# Napredni razvoj programske potpore za web

**- predavanja -**

**2022./2023.**

## Progresivne web-aplikacije

## PWAs

# Creative Commons

- slobodno smijete:
  - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
  - **prerađivati** djelo
- pod sljedećim uvjetima:
  - **imenovanje:** morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
  - **nekomercijalno:** ovo djelo ne smijete koristiti u komercijalne svrhe.
  - **dijeli pod istim uvjetima:** ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

*U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela.*
*Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.*
*Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.* *Tekst licence preuzet je s http://creativecommons.org/*

# Progressive Enhancement

- *Progressive enhancement is a **design philosophy** that provides a **baseline of essential content** and functionality to as many users as possible, while delivering the **best** possible **experience only** to users of the most **modern browsers** that can run all the required code.* https://developer.mozilla.org/en-US/docs/Glossary/Progressive_Enhancement
- Na starijim preglednicima svejedno uporabljivo
- Unaprjeđuje se korisničko iskustvo za korisnike sposobnijih/modernijih preglednika
- Uobičajene tehnike:
  - *Feature detection*
  - *Polyfills*
- Povezan pojam: *Graceful degradation* (isto načelo, suprotnih "smjer kretanja")

# PWAs

- ***Progressive Web Apps*** *(PWAs) are web apps that use service workers, manifests, and other web-platform features in combination with progressive enhancement to give users an experience* ***on par with native apps****.* https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps
- Omogućuju mnoge prednosti:
  - Installable -> Home screen
  - progressively enhanced
  - responsively designed
  - re-engageable →PUSH notifikacije
  - Linkable
  - discoverable
  - network independent -> Service workers, keširanje
  - secure

# Internet preglednik!

- PWA su itekako utemeljene na preglednicima
- Preglednici poput virtualnog stroja - apstrahiraju OS!
  - Postoje neke sličnosti s Electron radnim okvirom

# Kada i zašto, tko?

- Kada web-app možemo smatrati PWA?
  - Ne postoji jednoznačan odgovor, *Lighthouse*
- Kako ju prepoznati?
  - Teško, ali to i je ideja ☺
  - (slika na sljedećem slajdu)
- Tko koristi?
  - Mnogi:
    - https://www.tigren.com/examples-progressive-web-apps-pwa/
    - https://pairroxz.com/blog/progressive-web-app-examples/
- Je li 100% prihvaćeno?
  - Ne - Apple, Firefox, …

# Kako prepoznati?

- Npr., aliexpress

# PWA tehnologije

1. **Manifest**
2. **Service worker**
   - *Caching*
   - *Offline* rad
   - *Background sync*
   - *Push* notifikacije

- Često u kombinaciji s:
  - Media API
    - Kamera
    - Mikrofon
  - Geolocation API
    - Pozicija
  - Lokalna pohrana podataka
    - IndexedDB

# 1. Manifest (značajniji atributi)

```
{
    "name": "PWA-Cookbook-01",
    "short_name": "CookBook",
    "description": "Demo aplikacija u predmetu Napredni razvoj za web",
    "display": "fullscreen",
    "theme_color": "red",
    "background_color": "#DDD",
    "start_url": "/index.html",
    "scope": ".",
    "orientation": "portrait-primary",
    "dir": "ltr",
    "lang": "en-US",
    "icons": [
      {
        "src": "img/windows10/SmallTile.scale-100.png",
        "sizes": "71x71"
      },
      ...
```

- Standalone
- Fullscreen
- Minimal-UI
- Browser

*The background_color member defines a placeholder background color for the application page to display before its stylesheet is loaded. This value is used by the user agent to draw the background color of a shortcut when the manifest is available before the stylesheet has loaded.*

*The start_url member is a string that represents the start URL of the web application — the preferred URL that should be loaded when the user launches the web application (e.g., when the user taps on the web application's icon from a device's application menu or homescreen).*

*The scope member is a string that defines the navigation scope of this web application's application context. It restricts what web pages can be viewed while the manifest is applied. If the user navigates outside the scope, it reverts to a normal web page inside a browser tab or window.*
*If the scope is a relative URL, the base URL will be the URL of the manifest*

# PWA display modes

Standalone

Fullscreen

Minimal-UI

Browser

*fallback*

# Probajmo s emulatorom

- Android studio
- Tools -> AVD manager
- Otvorimo 10.0.2.2 (a ne localhost, jer to bio sam pametni telefon)
- Chrome
  - Možemo (zasad samo) ručno dodati na Home screen

# Pogledajmo i devtools -> Application

VER: 01

Galaxy S5 ▾ 360 x 640 100% ▾

Elements Console Sources Network Performance Memory Application Security

**Cookbook PWA v01 (manifest)**

Application
- Manifest
- Service Workers
- Storage

Storage
- Local Storage
- Session Storage
- IndexedDB
- Web SQL
- Cookies
- Trust Tokens

Cache
- Cache Storage
- Application Cache

Background Services
- Background Fetch
- Background Sync
- Notifications
- Payment Handler
- Periodic Background Sync
- Push Messaging

Frames
- top

App Manifest

manifest.json

**Installability**

⚠ No matching service worker detected. You may need to reload the page, or check that the scope of the service worker for the current page encloses the scope and start URL from the manifest.

**Identity**

| | |
|---|---|
| Name | PWA-Cookbook-01 |
| Short name | CookBook |
| Description | Demo aplikacija na predmetu Napredni razvoj za web |

**Presentation**

| | |
|---|---|
| Start URL | /index.html |
| Theme color | red |
| Background color | #DDD |
| Orientation | portrait-primary |
| Display | fullscreen |

**Icons**

☐ Show only the minimum safe area for maskable icons

Need help? Read documentation on maskable icons.

Primary icon as used by

Chrome

71×71px

# Installability

- Navodno ažurni kriteriji za Chrome: https://developers.google.com/web/fundamentals/app-install-banners/native
  - nisu
- **Dodao:**
  - prefer_related_applications, related_application
  - **HTTPS** s ispravnim certifikatom
    - Rješenje – postaviti aplikaciju na neki besplatan hosting, npr. https://www.cloudsavvyit.com/5057/how-to-host-a-static-website-for-free-on-googles-firebase-hosting-platform/
    - https://pwa02-92063.web.app/
  - Service worker, registiran, fetch handler

# 2. Service worker

- **Izvanmrežni** (*offline*) rad
  - Keširanje
  - Možemo presretati ***fetch*** zahtjeve i sami odgovarati
- **Push** notifikacije
- Service worker je JS datoteka
  - Naš JS kod s kojim kontroliramo ponašanje
- **Izvodi se mimo glavne preglednikove UI dretve**
  - **ServiceWorkerGlobalScope**
  - **nemaju pristup DOM-u**
  - **`self` objekt**
- **Preduvjet – HTTPS**
  - Iznimka:
    http://localhost...

# Životni ciklus SW-a

- SW moramo registrirati
  - `navigator.serviceWorker.register('/sw.js')`
  - Slijedi dohvat i instalacija (ako već nije instaliran)
- Po uspješnoj instalaciji i aktivaciji (vidi sljedeći slajd) SW će **kontrolirati** klijente (*clients*) u svom opsegu (***scope***)
  - „We call pages, workers, and shared workers **clients**"
  - **Kontrolirati** = mrežni zahtjevi idu preko SW-a; push notifikacije
  - Možemo provjeriti tko kontrolira s:
    `navigator.serviceWorker.controller` (null ili SW instanca)
  - Defaultni i maksimalni scope je **./** od URL-a skripte
    - https: *//www.fer.unizg.hr/je/super/sw.js*
      -> *https: //www.fer.unizg.hr/je/super*

# Životni ciklus *service workera*

- Aktivan kada je spreman da kontrolira klijente
- *Ne znači da će kontrolirati onoga koji je pokrenuo register*

- **Samo jednom**
- **Install event**
- S `installEvent.waitUntil()` možemo sačekati (promise) i provjeriti uspjeh instalacije
- Fetch i push idu mimo workera dok ne postane aktivan, ali ni tada ako nije stranica otvorena kroz SW
  - refresh
  - Ili `clients.claim()`

*"document starts life with or without a Service worker and maintains that for its lifetime"*

https://www.digitalocean.com/community/tutorials/demystifying-the-service-worker-lifecycle

https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API/Using_Service_Workers

# Cats and dogs – index.html

- Primjer preuzet s: https://developers.google.com/web/fundamentals/primers/service-workers/lifecycle
- SW presreće zahtjev i umjesto psa vraća mačku (slj. slajd)
  - Ali: treba refresh da počne kontrolirati

```
<!DOCTYPE html>
<h1>An image will appear here in 3 seconds:</h1>
<script>
    navigator.serviceWorker.register('./sw.js')
        .then(reg => console.log('SW registered!', reg))
        .catch(err => console.log('Boo!', err));

    setTimeout(() => {
        const img = new Image();
        img.src = './dog.svg';
        document.body.appendChild(img);
    }, 3000);
</script>
```

```
SW registered!                                                    (index):5
▸ ServiceWorkerRegistration {installing: ServiceWorker, waiting: null, active: null, navigationPreload: NavigationPreloadManager, scope: 'http
  s://service-worker-lifecycle-demos.glitch.me/main/', …}
V1 installing…                                                    sw.js:2
V1 now ready to handle fetches!                                   sw.js:10
> |
```

# Cats and dogs – sw.js

- Puno novih stvari: cache API, self, SW events…

```
self.addEventListener('install', event => {
    console.log('V1 installing...');
    event.waitUntil( // cache a cat SVG
        caches.open('static-v1').then(cache => cache.add('./cat.svg'))
    );
});
self.addEventListener('activate', event => {
    console.log('Activated, V1 now ready to handle fetches!');
});
self.addEventListener('fetch', event => {
    const url = new URL(event.request.url);
    // serve the cat SVG from the cache if the request is
    // same-origin and the path is '/dog.svg'
    if (url.origin == location.origin && url.pathname == '/dog.svg') {
        event.respondWith(caches.match('/cat.svg'));
    }
});
```

https://developer.mozilla.org/en-US/docs/Web/API/CacheStorage

Što keširati? CSS; JS; slike, itd,

# Pogledajmo SW u devtools:

# Ažuriranje SW-a

- Potrebno je promijeniti barem jedan bajt da se SW smatra novom verzijom
  - Mijenjanje URL-a SW-a nije dobra praksa
- ALI: novi SW će se instalirati ali neće postati aktivan dok barem jedna stranica (tab) koristi stari SW!
  - To znači da postoje dva SW-a: jedan (stari) aktivan i novi koji čeka
- Potrebno je:
  - Zatvoriti sve stranice/tabove da bi novi SW postao aktivan
  - Ili programski forsirati sa `self.skipWaiting()`
- U razvojnoj okolini puno lakše - **devtools**:
  - **update on reload**
  - Unregister/register
  - Skip waiting
- Detaljnije na: https://developers.google.com/web/fundamentals/primers/service-workers/lifecycle#updates

# Cache API

- Key-value baza
- Može se koristiti i iz „običnog” JS-a

# Što keširati?

- Slike, CSS, JS, fontovi, …
- Primijetiti:
  - **Statički** sadržaj (keširati kod **instalacije**, ***app shell***)
  - **Dinamički** sadržaj
    - Keširati (ako ima smisla) za vrijeme rada aplikacije
- Keširanje kostura aplikacije (*app shell cacheing*):
  - Isključimo mrežu i pogledamo kako nam aplikacija izgleda
  - Popravimo, ponovimo

VER: 04

# App shell + dinamičko keširanje

- Ovdje ćemo pokazati raskošnije rješenje koje:
  - **Djelomično** (popraviti – kako ih sve odrediti?) kešira app shell zahtjeve **kod instalacije**
  - **Dinamički kešira** sve ostale zahtjeve **kako zahtjevi dolaze**, te je u konačnici sve keširano
  - Uvodimo **404 stranicu** koju keširamo za slučaj 404
  - Uvodimo **Offline stranicu** koju možemo prikazati ako nismo keširali neku stranicu.

# SW offline - instalacija

VER: 04

```
const filesToCache = [
    "/",
    "manifest.json",
    "index.html",
    "offline.html",
    "404.html",
  "https://fonts.googleapis.com/css2?family=Fira+Sans:ital,wght@0,400;0,700;1,400;1,700&display=swap",
  "https://fonts.gstatic.com/s/firasans/v11/va9E4kDNxMZdWfMOD5Vvl4jLazX3dA.woff2",
  "https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css",
];

const staticCacheName = "static-cache-v1";

self.addEventListener("install", (event) => {
    console.log("Attempting to install service worker and cache static assets");
    event.waitUntil(
        caches.open(staticCacheName).then((cache) => {
            return cache.addAll(filesToCache);
        })
    );
});
```

Popis **zahtjeva** (stringovi ne temelju kojih će se napraviti zahtjevi) koje ćemo inicijalno (kod **instalacije**) keširati

`event.waitUntil(PROMISE)`

Inače bi instalacija završila dok još nije keširano

Obavlja jedan po jedan request na zadani URL i kešira: `(request, response)`

# SW offline - fetch

VER: 04

```
self.addEventListener("fetch", (event) => {
  event.respondWith(
    caches
      .match(event.request)
      .then((response) => {
        if (response) {
          return response;
        }
        return fetch(event.request).then((response) => {
          if (response.status === 404) {
            return caches.match("404.html");
          }
          return caches.open(staticCacheName).then((cache) => {
            cache.put(event.request.url, response.clone());
            return response;
          });
        });
      })
      .catch((error) => {
        return caches.match("offline.html");
      })
  );
});
```

Pretražuje sve cacheve po ključu event.request

Pronašli u *cacheu*, vraćamo spremljeni odgovor

SW dohvaća s interneta

Potencijalno vraćamo spremljenu 404 stranicu

„Clone is needed because put() consumes the response body.”

Prvo pohranjujemo odgovor u cache, te konačno vraćamo odgovor.
Sljedeći put bi morao biti u cacheu.
***Je li ova strategija dobra - sve spremamo u isti cache i nakon nekog vremena više uopće ne idemo na internet?***

# Bolji osnovni pristup

- Napraviti dva cachea (keys):
  - STATIC_CACHE_V<N>
  - DYNAMIC_CACHE
- Kod instalacije staviti app shell u statički cache
- Prilikom rada staviti datoteke (ne i JSON zahtjeve!) u dinamički cache
- Kod nove verzije SW-a, promijeniti verziju statičkog cachea, te:
  - Prilikom **aktivacije** obrisati sve druge (stare cacheve)
  - ✓ Dok novi SW nije aktiviran stari cachevi su još prisutni jer ih možda koriste druge aplikacije

```
self.addEventListener("activate", (event) => {
  const cacheWhitelist = [staticCacheName];
  event.waitUntil(
    caches.keys().then((cacheNames) => {
      return Promise.all(
        cacheNames.map((cacheName) => {
            if (cacheWhitelist.indexOf(cacheName) === -1) {
                return caches.delete(cacheName);
            }
        })
      );
    })
  );
});
```

# Strategije keširanja

- Postoje brojne strategije keširanja, najpoznatije su:
  - ***Stale-While-Revalidate***

  - ***Cache First (Cache Falling Back to Network)***

  - ***Network First (Network Falling Back to Cache)***

  - ***Network Only***

  - ***Cache Only***

- Primijetiti da možemo kombinirati strategije za različite skupine datoteka/zahtjeva

https://developer.chrome.com/docs/workbox/caching-strategies-overview/

# Keširanje pomoću workboxa

- Googleov alat koji uvelike pomaže razvoju SW-a
- Ugrađene strategije (s prethodnog slajda)
- Config -> generate -> sw.js
- Opaska: za cacheiranje podataka (JSON i sl.) bolje koristiti IndexedDB, a ne Cache Storage

```
import {ExpirationPlugin} from 'workbox-expiration';
import {registerRoute} from 'workbox-routing';
import {StaleWhileRevalidate} from 'workbox-strategies'; // Cache Google Fonts with a stale-while-revalidate strategy, with max num of entries.
registerRoute(
  ({url}) => url.origin === 'https://fonts.googleapis.com' ||
             url.origin === 'https://fonts.gstatic.com',
  new StaleWhileRevalidate({
    cacheName: 'google-fonts',
    plugins: [
      new ExpirationPlugin({maxEntries: 20}),
    ],
  }),
);
```

https://developers.google.com/web/tools/workbox

# Pozadinska sinkronizacija

*Background sync*

# Pozadinska sinkronizacija (*background sync*)

```
// Register sync - zatražimo jednokratnu sinkronizaciju
navigator.serviceWorker.ready.then(function(swRegistration) {
  return swRegistration.sync.register('myFirstSync');
});
```

```
self.addEventListener('sync', function (event) {
  if (event.tag == 'myFirstSync') {
    event.waitUntil(posaljiPodatke());
  }});
```

Radi i ako "zatvorimo tab"!

# IndexedDB

| API | Data Model | Persistence | Browser Support | Transactions | Sync/Async |
|---|---|---|---|---|---|
| **File system** | Byte stream | device | 52% | No | Async |
| **Local Storage** | key/value | device | 93% | No | Sync |
| **Session Storage** | key/value | session | 93% | No | Sync |
| **Cookies** | structured | device | 100% | No | Sync |
| **Cache** | key/value | device | 60% | No | Async |
| **IndexedDB** | **hybrid** | **device** | **83%** | **Yes** | **Async** |

https://blog.sessionstack.com/how-javascript-works-storage-engines-how-to-choose-the-proper-storage-api-da50879ef576

- Nažalost: API je prilično nezgrapan
- Nasreću:
  - https://github.com/jakearchibald/idb
  - https://www.npmjs.com/package/idb-keyval (trivijalni API)

# Plan

"prljavi trik": radi jednostavnosti nećemo koristiti bazu podataka već ćemo naziv (title) slike staviti u filename, tako da istovremeno pohranimo i sliku i title

# Kako fotografirati?

- Native API
  - Navigator.mediaDevices
  - Preko preglednika pristupamo kameri uređaja, bilo da je na pametnom telefonu, laptopu ili webcam na stolnom računalu
- "Trik": player -> canvas
  - Prikazan je ILI player ILI canvas
  - Konačno, iz canvasa generiramo png i pohranimo u IndexedDB

```
<video id="player" width="100%" autoplay>
</video>
navigator.mediaDevices
  .getUserMedia({ video: true,
                  audio: false })
  .then((stream) => {
     player.srcObject = stream;
  }).catch((err) => {...});
```

```
<canvas id="cnvFood"></canvas>
</video>
canvas
  .getContext("2d")
  .drawImage(player, 0, 0,
             canvas.width, canvas.height);
```

# brag.html (1/2)

VER: 05

```
import { get, set } from "https://cdn.jsdelivr.net/npm/idb-keyval@6/+esm";
let player = document.getElementById("player");
let canvas = document.getElementById("cnvFood");
let beforeSnap = document.getElementById("beforeSnap");
let afterSnap = document.getElementById("afterSnap");
let snapName = document.getElementById("snapName");
let startCapture = function () {
    beforeSnap.classList.remove("d-none");
    afterSnap.classList.add("d-none");
    if (!("mediaDevices" in navigator)) {
        // fallback to file upload button, ili sl.
        // vidjet i custom API-je: webkitGetUserMedia i mozGetU
    } else {
        navigator.mediaDevices
            .getUserMedia({ video: true, audio: false })
            .then((stream) => {
                player.srcObject = stream;
            })
            .catch((err) => {
                alert("Media stream not working");
                console.log(err);
            });
    }
};
startCapture();
```

Sakrijemo canvas, pokažemo player

U PE/GD smislu bi trebalo obraditi preglednike koji ne podržavaju mediaDevices i omogućit button za file upload

Video stream pridružujemo video elementu

Pokrećemo, čim se otvori stranica brag.html

# brag.html (2/2)

VER: 05

```
document.getElementById("btnSnap").addEventListener("click", function (event) {
    canvas.width = player.getBoundingClientRect().width;
    canvas.height = player.getBoundingClientRect().height;
    canvas.getContext("2d")
        .drawImage(player, 0, 0, canvas.width, canvas.height);
    stopCapture();
});
document.getElementById("btnUpload") addEventListener("click", function (event) {
    ...
    if ("serviceWorker" in navigator && "SyncManager" in window) {
        fetch(canvas.toDataURL())
            .then((res) => res.blob())
            .then((blob) => {
                let ts = new Date().toISOString();
                let id = ts + snapName.value.replace(/\s/g, "_"); // ws->_
                set(id, { id, ts, title: snapName.value, image: blob });
                return navigator.serviceWorker.ready;
            }).then((swRegistration) => {
                return swRegistration.sync.register("sync-snaps");
            }).then(() => {
                console.log("Queued for sync");
                startCapture();
            }).catch((err) => { console.log(error); });
    } else { // fallback: pokusati poslati, pa ako ima mreze onda dobro...
        alert("TODO - vaš preglednik ne podržava bckg sync...");
    }
});
```

Kopiramo sadržaj playera u canvas i zaustavljamo stream i skrivamo player tj. otkrivamo canvas

Pretvaramo Base64 kodiranu sliku s canvasa u blob, te ju pohranjujemo u IndexedDB

Registriramo jednokratni sync event. Ako je mreža dostupna, sync event nad SW-om će biti odmah okinut.

# sw.js

VER: 05

```
self.addEventListener("sync", function (event) {
  if (event.tag === "sync-snaps") {
    event.waitUntil(syncSnaps());
  }
});
let syncSnaps = async function () {
  entries().then((entries) => {
    entries.forEach((entry) => {
      let snap = entry[1]; //  Each entry is an array of [key, value].
      let formData = new FormData();
      formData.append("id", snap.id);
      formData.append("ts", snap.ts);
      formData.append("title", snap.title);
      formData.append("image", snap.image, snap.id + ".png");
      fetch("/saveSnap", {
        method: "POST",
        body: formData,
      }).then(function (res) {
          if (res.ok) {
              res.json().then(function (data) {
                  console.log("Deleting from idb:", data.id);
                  del(data.id);
              });
          } else {console.log(res);}
      }).catch(function (error) {console.log(error);});
    });
  });
};
```

Obrađujemo naš problem

idb-keyval na ovaj način dohvaća sve zapise iz IndexedDB baze

Sklapamo i šaljemo POST zahtjev koji uključuje i našu sliku. Podsjetnik: koji je content-type ovog zahtjeva?

Ako smo uspjeli server.js će nam vratiti nazad id koji smo poslali koji sad koristimo da obrišemo zapis iz IndexedDB

# ZOKŽZV: Periodička pozadinska sinkronizacija

- Nalik "običnoj" ali se periodički ponavlja u zadanom intervalu, npr: https://developer.mozilla.org/en-US/docs/Web/API/Web_Periodic_Background_Synchronization_API

```
async function registerPeriodicNewsCheck() {
  const registration = await navigator.serviceWorker.ready;
  try {
    await registration.periodicSync.register('get-latest-news', {
      minInterval: 24 * 60 * 60 * 1000,
    });
  } catch { console.log('Periodic Sync could not be registered!'); }
}
```

```
self.addEventListener('periodicsync', event => {
  if (event.tag == 'get-latest-news') {
    event.waitUntil(fetchAndCacheLatestNews());
  }
});
```

# Push notifikacije

# Kako rade push notifikacije?

# Primijetiti – notifikacije su nezavisne od SW-a…

- … iako se najčešće koriste u kombinaciji sa SW-om
  - Kada tražimo dozvolu za notifikaciju, odmah dobijemo i implicitnu dozvolu za push
- Ali, npr. možemo koristiti i bez SW-a, npr.:
  - Google photos – pokrenemo backup i onda aplikacija radi backup u pozadini te nas obavijesti putem notifikacije kad je gotova
- Opcije:

https://developers.google.com/web/fundamentals/push-notifications/display-a-notification

- Demo: https://web-push-book.gauntface.com/demos/notification-examples/

# Plan

- Dodati gumb u index.html gdje se može pretplatiti na notifikacije
  - Istovremeno i push dozvola
- Spremiti pretplatu na server
  - Array (in memory)
  - Pohranjujemo u datoteku
- Kada se snimi slika poslati notifikaciju
- Prikazati notifikaciju u SW-u
  - OS prikazuje notifikaciju
  - Kada korisnik klikne, SW event
- Preusmjeriti klijente koje kontrolira SW na stranicu koja je zadana u notifikaciji (index.html)

https://floating-oasis-41640.herokuapp.com/

# VAPID ključevi, web-push

- **npm i web-push**
- web-push generate-vapid-keys
- Ili preko npm run:

```
λ cat package.json | grep gen
    "gen-vapid": "web-push generate-vapid-keys"

C:\Users\Igor\OneDrive - fer.hr\Nastava\Web2 - Napredni razvoj programske potpore za web\wip-p
λ npm run gen-vapid

> pwa-examples@1.0.0 gen-vapid
> web-push generate-vapid-keys


=======================================

Public Key:
BBfae6kt70vtdHKE_w3sd2c9viue80_wUXE6ZMjkRprWCtjQ5ZgzqGDWxkc79ncxc2LCSGuFnVNYrJSYX9NWG8Y

Private Key:
fIgVNhAHkYrhJXNWca_4qTaykZamA4Y1xdDGrXVAtxc

=======================================


C:\Users\Igor\OneDrive - fer.hr\Nastava\Web2 - Napredni razvoj programske potpore za web\wip-p
λ npm run gen-vapid

> pwa-examples@1.0.0 gen-vapid
> web-push generate-vapid-keys


=======================================

Public Key:
BMpYzuKrXO0AEP8h9dagpxkz6f4BLgbWjToFz8t228MWYlpccGHLxU8LM7f1y2X6sImk8aIBID0v-RVDqTwNmNI

Private Key:
AenZ17i7lAD8Mdi6QujHEEEpEQU1axXYF3dTgxb2KiM

=======================================
```

- Trebat će nam:
  - Na klijentu:
    - Public key
  - Na serveru:
    - Public key
    - Private key

# push.js (iz index.html)

VER: 06

```
if ("Notification" in window
  && "serviceWorker" in navigator) {
  btnNotif.addEventListener("click", function () {
    Notification.requestPermission(
     async function (res) {
      if (res === "granted") {
        await setupPushSubscription();
      } else {
        console.log("User denied push notifs:", res)
      }
    });
  });
} else {
  btnNotif.setAttribute("disabled", "");
  btnNotif.classList.add("btn-outline-danger");
}
```

```
async function setupPushSubscription() {
  try {
    let reg = await navigator.serviceWorker.ready;
    let sub = await reg.pushManager.getSubscription();
    if (sub === null) {
      let publicKey = "BL1oXiSXCjK(...)dT2EJGH33qe5iw";
      sub = await reg.pushManager.subscribe({
          userVisibleOnly: true,
          applicationServerKey: urlBase64ToUint8Array(publicKey)
      })
      let res = await fetch("/saveSubscription", {
         method: "POST", headers: {
         "Content-Type": "application/json",
         Accept: "application/json",
        }, body: JSON.stringify({ sub }),
      });
      if (res.ok) {
         alert("Yay, subscription generated a  saved:\n" +
            JSON.stringify(sub));
      }
    } else { alert("You are alreay subscribed
    } catch (error) {
       console.log(error);
    }
}
```

Javni VAPID ključ

Preglednik pita korisnika za dozvolu

Kontraktiramo preglednikov push service, ključ ne može ići plain-text, moramo ga prekodirati

Pohranjujemo pretplatu „kod sebe”

# server.js (pohrana pretplata i slanje notifikacija)

VER: 06

```
const webpush = require('web-push');
// Umjesto baze podataka, čuvam pretplate u
// datoteci:
let subscriptions = [];
const SUBS_FILENAME = 'subscriptions.json';
try {
  subscriptions =
JSON.parse(fs.readFileSync(SUBS_FILENAME));
} catch (error) {
  console.error(error);
}

app.post("/saveSubscription", function(req, res) {
  let sub = req.body.sub;
  subscriptions.push(sub);
  fs.writeFileSync(SUBS_FILENAME,
     JSON.stringify(subscriptions));
  res.json({
    success: true
  });
});
```

Poziva se iz već viđene saveSnaps

```
async function sendPushNotifications(snapTitle) {
  webpush.setVapidDetails('mailto:ime.prezime@fer.hr',
  'BL1oXi(...)T2EJGH33qe5iw',
  '4B9u(...)BZshEZnI');
  subscriptions.forEach(async sub => {
    try {
      await webpush.sendNotification(sub, JSON.stringify({
        title: 'New snap!',
        body: 'Somebody just snaped a new photo: ‘
               + snapTitle,
        redirectUrl: '/index.html'
       }));
    } catch (error) {
      console.error(error);
    }
  });
}
```

payload

# sw.js (⚡push ⚡notificationclick)

VER: 06

```
self.addEventListener("push", function (event) {
  var data = { title: "title", body: "body",
               redirectUrl: "/" };

  if (event.data) {
    data = JSON.parse(event.data.text());
  }
  let options = {
    body: data.body,
    icon: "assets/img/android/android-
launchericon-96-96.png",
    badge: "assets/img/android/android-
launchericon-96-96.png",
    vibrate: [200, 100, 200, 100, 200, 100, 200],
    data: {
      redirectUrl: data.redirectUrl,
    },
  };

  event.waitUntil(
    self.registration.showNotification(
      data.title, options)
  );
});
```

Na neki način, sami sebi šaljemo, u drugi event od SW-a

```
self.addEventListener("notificati[illegible]k", function (event) {
  let notification = event.noti[illegible]tion;
  // mogli smo i definirati [illegible]tons, pa ovdje granati s
  //obzirom na:  event.ac[illegible]on;
  event.waitUntil(
    clients.matchAll().then(function (clis) {
      clis.forEach((client) => {
        client.navigate(notification.data.redirectUrl);
        client.focus();
      });
      notification.close();
    })
  );
});

self.addEventListener("notificationclose", function (event) {
  console.log("notificationclose", event);
});
```

Vraća sve klijente koje kontrolira ovaj SW

Otvaramo "/index.html" koji smo zadali još u server.js

Postoji i `notificationclose`, ali ga ne koristimo u ovom scenariju

# Nekoliko tehničkih opaski

- Dominantno razvijamo na:
  - Chrome
  - http://localhost (jer ako je localhost onda ne mora biti https)
  - Samopotpisani certifikati i lokalni HTTPS server nam nisu od pomoći:
    - Neće raditi ni manifest install
    - Neće registrirati SW
- Kako onda testirati s mobilnog uređaja?
  - Free account na https://www.heroku.com/free
  - https://devcenter.heroku.com/articles/getting-started-with-nodejs
  - Dovoljno je imati http server:

```
const httpPort = process.env.PORT || 5000;
```

# Korisni izvori

- https://developers.google.com/web/ilt/pwa
- https://developers.google.com/web/tools/workbox
- https://github.com/hemanth/awesome-pwa
- https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps
- https://serviceworke.rs/

# Napredni razvoj programske potpore za web

**- predavanja -**

**2022./2023.**

## CSS

## (nastavak na W1)

# Creative Commons

- slobodno smijete:
  - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
  - **prerađivati** djelo
- pod sljedećim uvjetima:
  - **imenovanje:** morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
  - **nekomercijalno:** ovo djelo ne smijete koristiti u komercijalne svrhe.
  - **dijeli pod istim uvjetima:** ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

*U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela.*
*Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.*
*Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.* *Tekst licence preuzet je s http://creativecommons.org/*

# Većina obrađena u Web1

- Još par odabranih tema: transform i animacije

Taxonomy and status of CSS3 **modules**.

- Recommendation
- Candidate Recommendation
- Last Call
- Working Draft.

#Cascading_Style_Sheets

# Sadržaj

- CSS (nastavak):
  - Transform
  - Animations
- ----------

- CSS jezici
  - SASS/SCSS

- *CSS scope*
- *CSS frameworks*

# tranform

- Hardversko ubrzanje
- Elementi ostaju u toku (*flow*)
- **Glavne** opcije:
  - **Pomak** (možemo i s top, left, margin ali ovo je bolje)
    - `transform: translate(120px, 50%);`
  - **Skaliranje**
    - `transform: scale(2, 0.5);`
  - **Rotacija**, dodatna opcija: transform-origin (npr. center, top left, itd.)
    - `transform: rotate(0.5turn);`
  - **Iskrivljavanje**
    - `transform: skew(30deg, 20deg);`
- Poredak je bitan (**zdesna na lijevo**)
  - https://stackoverflow.com/questions/27635272/css3-transform-order-matters-rightmost-operation-first

# Transform demo

# Transition (and animation)

- Kada mijenjamo neko CSS svojstvo to se događa momentalno (npr. boja pozadine iz bijele u plavu)
- `transition: <property> <duration> <timing-function> <delay>;`
- `Transition` nam omogućuje da kontroliramo, tj. animiramo promjenu svojstva, možemo kontrolirati:
  1. Koje svojstvo se mijenja, te za svako od svojstava definirati:
     - Nisu sva svojstva upravljiva: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_animated_properties
  2. Koliko vremena traje promjena
  3. Acceleration/easing function - funkciju promjene svojstava u vremenu (default - linearno)
     - Easing function cheat sheet: https://easings.net/
  4. Eventualni poček

# Transition demo

# Animations

- Ako trebamo još veću kontrolu nad promjenama možemo koristiti animacije
- Prvo, definirajmo keyframes (a zatim ih referenciramo), npr.:

```
@keyframes logo-path {
  from {
    background-color: red;
  }
  to {
    transform: translate(-50%, -50%) scale(1.3, 1.3) rotate(90deg);
    border: 3px solid green;
    background-color: cornflowerblue;
  }
}
```

Proizvoljno ime

Istog trena će prvo promijeniti boju pozadine u crvenu

Moguće je koristiti i postotke. from je isto kao 0% to 100%

# Animations

- Zatim referenciramo keyframes i podesimo parametre:
  - animation-name (ime keyframes)
  - animation-duration (trajanje)
  - animation-timing-function (acceleration/easing curve, isto)
  - animation-delay (poček)
  - animation-iteration-count (broj ponavljanja, moguće je infinite)
  - animation-direction (definira ponašanje nakon ciklusa)
  - animation-fill-mode (koje vrijednosti se primjenjuju prije i na kraju animacije - npr. želimo li da ostane u konačnom stanju?)
- Npr. animirani tekst iz sljedećeg primjer (animations demo)

```
#banner h2 {
  white-space: nowrap;
  animation-duration: 6s;
  animation-name: slidetext;
  animation-iteration-count: infinite;
  animation-timing-function: linear;
  animation-direction: alternate;
}
```

Mogao je i *shorthand*

```
@keyframes slidetext {
  from {
    margin-left: 100%;
  }
  to {
    margin-left: 0%;
  }
}
```

# Animations demo

# *Vendor prefixes*

- https://www.w3.org/Style/CSS/current-work
  - *Working draft, candidate recommendation (draft), candidate recommendation*
- Proizvođači preglednika žele omogućiti neka svojstva i prije nego što su u CR fazi
  - Definirati pomoću ***vendor prefix****a*
  - Čak i ako dođe do breaking changes neće doći do promjene u radu *early-adopting* web stranica
- Primjer na sljedećoj stranici

# Vendor prefixes - autoprefixer

- Napravimo c/p style sekcije iz npr. transition demo
  - https://autoprefixer.github.io/
  - https://developer.mozilla.org/en-US/docs/Glossary/Vendor_Prefix

```
#ferlogo:hover {
  -webkit-transform: translate(-50%,
                -50%) scale(1.3, 1.3) rotate(90deg);
  -ms-transform: translate(-50%,
                -50%) scale(1.3, 1.3) rotate(90deg);
  transform: translate(-50%,
                -50%) scale(1.3, 1.3) rotate(90deg);

  border: 3px solid green;
  background-color: cornflowerblue;
}
```

# ZOKŽZV

- pogledati:
  - @supports
    - https://developer.mozilla.org/en-US/docs/Web/CSS/@supports
  - polyfills

# Kako dalje?

- Ovime smo prošli "sva" značajnija CSS svojstva i module

- Kako dalje?
  - **Kako pisati / ustrojiti CSS u većim projektima?**
  - Kako pisati nazive klasa?
  - Koristiti ili ne CSS radne okvire?
- **Problem/svojstvo: CSS je globalan!**

Sass

Install Learn Sass Blog Documentation Get Involved Search

# CSS with superpowers

Sass is the most mature, stable, and powerful professional grade CSS extension language in the world.

Current Releases: Dart Sass 1.35.1 LibSass 3.6.5 Ruby Sass Implementation Guide

# SASS - Syntactically Awesome Style Sheets

- *CSS preprocessor* -> prevodi se u CSS
  - Preglednici ga ne poznaju, prevođenje u razvojnoj okolini
- Dvije sintakse (~ ekstenzije):
  - .sass ~ yml, python, koriste se novi red i uvlačenje (indent)
  - .scss - novija, "ista" kao CSS, blokovi s vitičastim zagradama
- Većina primjera u nastavku slijedi https://sass-lang.com/guide s malim izmjenama

# Varijable

- (u "međuvremenu" su podržane i u CSS-u)

```
λ sass -w .:.
Compiled site.scss to site.css.
Sass is watching for changes. Press Ctrl-C to stop.
```

site.scss

```scss
$brand-color: blue;
$font-heading: Arial, sans-serif;
$font-body: Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-body;
  color: $primary-color;
}
```

site.css

```css
body {
  font: 100% Helvetica, sans-serif;
  color: #333;
}
```

Namjerno dodan višak

# Gniježđenje

- HTML omogućuje gniježđenje (hijerarhijsku strukturu) - CSS ne!

- SASS omogućuje gniježđenje
  - Selektora
  - Svojstava

nesting.scss

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }
}
body {
  font: {
    family: Iosevka, sans-serif;
    size: 16px;
    weight: bold;
  }
}
```

Svojstvo font-

nesting.css

```
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}

body {
  font-family: Iosevka, sans-serif;
  font-size: 16px;
  font-weight: bold;
}
```

# Partials and modules

Moguće i u CSS-u, ali onda imamo više datoteka

- DRY principle
- Moguće je kod izdvojiti u posebne datoteke koje zatim možemo uključiti u drugim datotekama i tako ostvariti modularni ustroj SASS projekta
  - *partials*, počinju s _, ne prevode se zasebno
  - *modules*, prevode se zasebno
- Nalik include u C-u, nije problem ako se više puta uključi (kao i u C-u)
- Npr, mogli bi imati:
  - _colors.scss i _typography.scss
- …i onda ih uključivati gdje nam trebaju s @use (prije @import):
  - @use "colors";  @use "typography"

Ne treba navesti _ ni ekstenziju

# SASS-3-modules-and-partials

Primijetiti da nema colors.css jer je partial.

_colors.scss

```sass
$zelena: green;
$crvena: red;
$plava: blue;
```

typography.scss

```sass
body {
  font: {
    family: Iosevka;
    size: 16px;
    weight: bold;
  }
}
```

modules-partials.scss

```sass
@use "colors";
@use "typography";

body {
    color: colors.$plava;
}
```

modules-partials.css

```css
body {
  font-family: Iosevka;
  font-size: 16px;
  font-weight: bold;
}

body {
  color: blue;
}
```

# Naslijeđivanje

- Pomoću @extend možemo naslijediti (preuzeti) skup CSS svojstava od nekog selektora
  - Dodatno u primjeru koristimo "placeholder class" koja se materijalizira samo ako je naslijeđena - počinje s %

inheritance.scss

```scss
%button-default {
  font-size: 16px;
  cursor: pointer;
}
%ridiculous-button {  font-size: 66px; }

.button-success {
  @extend %button-default;
  background-color: green;
}
.button-danger {
  @extend %button-default;
  background-color: red;
}
```

Nije naslijeđen

inheritance.css

```css
.button-danger, .button-success {
  font-size: 16px;
  cursor: pointer;
}

.button-success {
  background-color: green;
}

.button-danger {
  background-color: red;
}
```

ZOKŽZV: https://csswizardry.com/2014/11/when-to-use-extend-when-to-use-a-mixin/

# SASS - itd.

- Svojstva:
  - **Variables**
  - **Nested selectors**
  - **Nested properties**
  - **Partials and modules, ~~@import~~/@use**
  - **media queries, nested**
  - **Inhertiance @extend**
  - Built-in modules (korisne funkcije: https://sass-lang.com/documentation/modules )
    - Lists, maps, npr. map of colors, math, …
  - Simple arithmetics https://sass-lang.com/documentation/operators
  - Mixins https://sass-lang.com/documentation/at-rules/mixin https://csswizardry.com/2014/11/when-to-use-extend-when-to-use-a-mixin/
  - Itd.
- Vrlo slično: LESS
  - https://lesscss.org/
  - https://css-tricks.com/sass-vs-less/

# Kako pisati CSS

- Sljedećih nekoliko slajdova preuzeto s: https://adamwathan.me/css-utility-classes-and-separation-of-concerns/

# 1. "Semantic" CSS

- "Separation of concerns"
  - HTML definira **sadržaj**
  - CSS definira **izgled**
- **Primjer: http://www.csszengarden.com/**

```html
<p class="text-center">
    Hello there!
</p>
```

```html
<style>
.greeting {
    text-align: center;
}
</style>

<p class="greeting">
    Hello there!
</p>
```

**Primjer s:**

https://adamwathan.me/css-utility-classes-and-separation-of-concerns/

**ALI: u složenijim slučajevima CSS nerijetko zrcali strukturu HTML-a**

# 2. Razdvajanje stilova od strukture

- Koristiti neku metodologiju imenovanja, npr. SMACSS ili BEM, vidjeti i npr. https://css-tricks.com/css-style-guides/
- BEM:

može i --

**`block__element--modifier`**

- Primjer na sljedećoj stranici ->
- Niska specifičnost (*low specificity*), izravnavanje specifičnosti selektora (*specificity flatness*)
- "*BEM gives everyone on a project a declarative syntax that they can share so that they're on the same page.*"
- https://css-tricks.com/bem-101/
- Problem održavanja koda - strah od izmjene

# http://getbem.com/introduction/

## Block

Standalone entity that is meaningful on its own.

### Examples

`header` , `container` , `menu` , `checkbox` , `input`

## Element

A part of a block that has no standalone meaning and is semantically tied to its block.

### Examples

Normal button | Success button | Danger button

`title`

## Modifier

A flag on a block or element. Use them to change appearance or behavior.

### Examples

`disabled` , `highlighted` , `checked` , `fixed` , `size big` , `color yellow`

We can have a normal button for usual cases, and two more states for different ones. Because we style blocks by class selectors with BEM, we can implement them using any tags we want ( `button` , `a` or even `div` ). The naming rules tell us to use `block--modifier-value` syntax.

### HTML

```
<button class="button">
        Normal button
</button>
<button class="button button--state-success">
        Success button
</button>
<button class="button button--state-danger">
        Danger button
</button>
```

### CSS

```
.button {
        display: inline-block;
        border-radius: 3px;
        padding: 7px 12px;
        border: 1px solid #D5D5D5;
        background-image: linear-gradient(#EEE, #DDD);
        font: 700 13px/18px Helvetica, arial;
}
.button--state-success {
        color: #FFF;
        background: #569E3D linear-gradient(#79D858, #569E3D) repeat-x;
        border-color: #4A993E;
}
.button--state-danger {
        color: #900;
}
```

# 3. Nezavisni CSS s gotovim komponentama

- CSS definira stil standardnih komponenti
  - **.button**
  - **.card**
  - **itd.**
- **Primjer - Bootstrap**

```
<div class="card" style="width: 18rem;">
  <img src="..." class="card-img-top" alt="...">
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">Some quick example text
to build on the card title and make up the bulk
of the card's content.</p>
    <a href="#" class="btn btn-primary">Go
somewhere</a>
  </div>
</div>
```

# 4. Nezavisni CSS s pomoćnim klasama

- CSS definira klase za
  - Text sizes, colors, and weights
  - Border colors, widths, and positions
  - Background colors
  - Flexbox utilities
  - Padding and margin helpers, …
- Primjer – TailwindCSS, Tachyons
- ***Enforced consistency***
  - Sužen skup mogućnosti
  - Npr. GitLab: 402 text colors, 239 background colors, 59 font sizes
- Treba li svejedno definirati komponente?

```
<button class="bg-blue-500
   hover:bg-blue-700
   text-white font-bold
   py-2 px-4 rounded">
  Button
</button>
```

Button

https://v1.tailwindcss.com/components/buttons

# Tko o kome ovisi?

- ...i što nam više odgovara?

1. HTML-u se može "izvana" promijeniti stil, ali CSS nije ponovo upotrebljiv
2. CSS ponovo upotrebljiv, HTML-u se ne može izvana promijeniti stil

# Radni okviri

- Dvije kategorije:
  1. **Sveobuhvatni** radni okviri, component frameworks koji sadrže i komponente i pomoćne klase, layout sustav, itd.
     - Npr. Bootstrap, Foundation, …
  2. Samo **pomoćne klase** (*utility classes*)
     - Npr. TailwindCSS, Tachyons, …

# Vanilla CSS ili radni okviri?

- https://academind.com/tutorials/vanilla-css-vs-frameworks/

| Vanilla CSS | Utility Frameworks | Component Frameworks |
|---|---|---|
| Full Control | Faster Development | Rapid Development |
| No unnecessary Code | Follow Best Practices | Follow Best Practices |
| Name Classes as you like | No Expert Knowledge Needed | No Need to be an Expert |
| Build everything from Scratch | Little Control | No or Little Control |
| Danger of “bad code” | Unnecessary Overhead Code | Unnecessary Overhead Code |
| | | “All Websites Look the Same” |

# Nove „komplikacije” odnosno mogućnosti

- **SPA** radni okviri koji imaju **lokalni CSS scope**, na razini komponenti
  - U sljedećim predavanjima
- *Cascade layers*
  - Proposal
  - https://css-tricks.com/cascade-layers/

# Napredni razvoj programske potpore za web

**- predavanja -**

**2022./2023.**

## 7. Jednostranične web-aplikacije

## (1/3)

# Creative Commons

- slobodno smijete:
  - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
  - **prerađivati** djelo
- pod sljedećim uvjetima:
  - **imenovanje:** morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
  - **nekomercijalno:** ovo djelo ne smijete koristiti u komercijalne svrhe.
  - **dijeli pod istim uvjetima:** ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

*U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela.*
*Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.*
*Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.* *Tekst licence preuzet je s http://creativecommons.org/*

# Podsjetimo se – tri arhitekture web-aplikacija

## 1. Statičke web-stranice

- ✓ **Performanse**
- ✓ Neovisnost o platformi
- ✓ Sigurnost
- ✗ „Statičnost”, klijent mora osigurati bilo kakvu interaktivnost

# Podsjetimo se – tri arhitekture web-aplikacija

## 2. Dinamičke web-stranice (generirane na poslužitelju)

- ✓ **Dinamički sadržaj, prilagođen klijentu**
- ✓ Sigurnost, odabir tehnologije, …
- ✗ Procesiranje (opterećenje) na poslužitelju
- ✗ Responzivnost (UX): klijent predaje podatke i **čeka** odgovor

# 3. Jednostranične web-aplikacije

- **SPA:** ***Single Page web Applications***

✓ **Dinamički sadržaj, prilagođen klijentu**

✓ **Performanse**: dobar dio opterećenja se prebacuje na klijenta

✓ **UX** – doima se kao (mobilna) aplikacija

✗ Kompleksnost, teže testiranje, sigurnost, zadan JS

# Kao i obično, ideja nije nova…

- ***Java applets***
  - https://en.wikipedia.org/wiki/Java_applet
  - Java applets were introduced in the first version of the Java language, which was released in **1995**.
  - Beginning in 2013, major web browsers began to phase out support for the underlying technology applets used to run, with applets becoming completely **unable to be run by 2015–2017**.
- ***ActiveX***
  - https://en.wikipedia.org/wiki/ActiveX
  - Microsoft introduced ActiveX in **1996**
  - ActiveX is still supported as of Windows 10 through Internet Explorer 11, while ActiveX is not supported in their default web browser Microsoft Edge
- ***Flex/Flash***
  - https://en.wikipedia.org/wiki/Apache_Flex
  - In November **1996**, FutureSplash was acquired by Macromedia, and Macromedia re-branded and released FutureSplash Animator as Macromedia Flash 1.0.
  - Adobe donated Flex to the Apache Software Foundation in 2011[2] and it was promoted to a top-level project in December 2012.
- ***Silverlight***
  - https://en.wikipedia.org/wiki/Microsoft_Silverlight
  - 2007-2011
- ***Itd, …razni JS kalkulatori, …***

- Novo: **WASM** (https://en.wikipedia.org/wiki/WebAssembly )

As of March 2021, less than 0.01% of sites used Silverlight,[24] 2.1% used the discontinued Adobe Flash,[25] and less than 0.01% use Java (client-side; server-side 3.4% use Java).[26][27]
https://en.wikipedia.org/wiki/Microsoft_Silverlight

# Primjer: µSPA

## Izgradimo vlastiti mikro radni okvir za jednostranične web-aplikacije

# Zašto radimo svoj FWK koji ćemo baciti u smeće?

- Upoznati:
  - načela rada
  - mehanizme i tehnologiju
  - probleme, vidjeti što je teško
  - razumjeti
  - Možda i naučiti malo JS-a u procesu ☺
- Iz istog razloga što:
  - Učimo zbrajati i množiti na papiru
  - Učimo pokazivače u C-u da bi shvatili *by value, by reference*
  - Itd.

# src

- `git clone git@gitlab.com:fer-web2/lectures/src.git`

# Osnovno načelo: mijenjamo DOM iz JS koda

ver1

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CookBook</title>
</head>
<body>
    <div id="uspa-app"></div>
</body>
<script>
    document
      .getElementById("uspa-app")
      .innerHTML = "<h1>My cookbook in the making...</h1>";
</script>
</html>
```

# Digresija: server.js

- FWK razvijamo kroz 4 verzije – jedan poslužitelj koji poslužuje sve 4 verzije na izbor

```
const getDirectories = source =>
    readdirSync(source, { withFileTypes: true })
        .filter(dirent => dirent.isDirectory() && dirent.name.startsWith("ver"))
        .map(dirent => dirent.name);
const arrVersionDirs = getDirectories(__dirname);
```

u tekućem direktoriju pronalazi sve direktorije koji počinju s "ver"

```
app.get('/', function (req, res) {
    res.send(`Please select version: <ul>${arrVersionDirs.map(name => `<li><a href="/${name}">${name}</a></li>`).join("")} </ul>`);
});
```

ver2

# Ver2: izmjestimo kod u modul

uspa > ver02 > assets > index.html > html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CookBook</title>
</head>
<body>
    <div id="uspa-app">
    </div>
</body>
<script type="module" src="/ver02/assets/js/uspa.js"></script>
</html>
```

novi ustroj projekta

ES modules
https://en.wikipedia.org/wiki/ECMAScript#6th_Edition_%E2%80%93_ECMAScript_2015

ver2

# Ver2: presrećemo svaki klik!

```
let template = `
<h1>My cookbook in the making ver2 (from the module)...</h1>
Pick one:
  <ul>
    <li><a href="http://fer.unizg.hr">Home</a></li>
    <li><a data-link href="http://fer.unizg.hr">Abroad</a></li>
  </ul>
`;

document.addEventListener("DOMContentLoaded", () => {
    document.body.addEventListener("click", e => {
        if (e.target.matches("[data-link]")) {
            e.preventDefault();
            document.getElementById("uspa-app")
                  .innerHTML = "Nemoj sine nikud ići...";
        }
    });
    document.getElementById("uspa-app").innerHTML = template;
});
```

# Ver2: presrećemo klik

ver2

- Pomalo pretjerano, presrećemo svaki klik, pa onda gledamo zanima li nas
  - Mogli smo npr. pronaći samo a elemente pa premostiti njihov `click` (isprobajte)

# µSPA - ver3

ver3

- Znamo:
  - Dinamički promijeniti sadržaj stranice
  - Presresti (i promijeniti) klik, npr. kod linkova
- Napravimo:
  - **Višestraničnu** web-aplikaciju – sami upravljamo promjenom stranica!
    - **routing**
  - Pritom preustrojimo kod tako da je svaka stranica poseban ***view*** odnosno posebna datoteka/modul
    - App view (defaultni view, aplikacija)
    - Recipes view (pregled recepata)
    - Calculator view (preračunavanje C <-> F (u ver4))

# Novi ustroj projekta

ver3

start.js

```
import Uspa from "./uspa.js"
import App from "./App.js"

const app = new Uspa();

app.mount("my-app", new App());
```

FWK je sad posebna datoteka tj. modul

# Svi viewovi moraju imati async getHtml()

ver3

App.js

```
import Recipes from "./views/Recipes.js";
import Calculator from "./views/Calculator.js";
export default class  {
    getViews() {
        return {
            Recipes,
            Calculator
        };
    }
    async getHtml() {
        return `
        <div class="d-flex justify-content-between bg-light border border-bottom border-primary p-3">
            <h1>My cookbook in the making ver3...</h1>
            <h2 class="d-flex">
                <ul class="d-flex ml-5 "  style="list-style: none;">
                    <li class="mr-3"><a data-link href="Recipes">Recipes</a></li>
                    <li class="mr-3"><a data-link href="Calculator" >Calculator</a></li>
                </ul>
            </h2>
        </div>
        <router-view></router-view>
        `;
    }
}
```

Calculator.js

```
export default class  {
    async getHtml() {
        return `
            <h1>Calc view</h1>
        `;
    }
}
```

# Uspa.js – nova verzija radnog okvira

ver3

uspa.js

```
export default class Uspa {
    async setView(viewName) {
        let viewClass = this.currView.getViews()[viewName];
        document.querySelector("router-view").innerHTML = await (new viewClass()).getHtml();
        history.pushState(null, null, `${this.stubUrl}/${viewName}`);
    }
    mount(selector, initalView) {
        this.currView = initalView;
        this.stubUrl = window.location.pathname;  // this is only because of server-side ver00 magic
        initalView.getHtml()
            .then(html => {
                document.getElementById(selector).innerHTML = html;
                return html;
            }).then(html => {
                if (html.indexOf("data-link" > 0)) {
                    document.body.addEventListener("click", e => {
                        if (e.target.matches("[data-link]")) {
                            e.preventDefault();
                            let viewName = e.target.getAttribute("href");
                            this.setView(viewName);
                        }
                    });
                }
            }).catch(err => {
                console.log("USPA: error occured:", err);
            });
    }
}
```

History API

```
...
app.mount("my-app", new App());
```

Npr. „Calculator”

# Višestranična web-aplikacija

ver3

- Sve se odvija na klijentu, aplikacija je responzivnija
- Mijenja se URL (s konačnim ciljem: *bookmarking*)
  - Routing nije dalje razvijan, koga zanima kompletniji primjer može pogledati https://www.youtube.com/watch?v=6BozpmSjk-Y&ab_channel=dcode ili https://blog.jeremylikness.com/blog/build-a-spa-site-with-vanillajs/

DZ: vizualno označite trenutno odabranu stanicu

# Recipes view

ver3

- Dohvatiti ćemo recepte s poslužitelja, formatirati i vratiti HTML
- Sad je jasno zašto getHtml() mora biti **async**
- **Uobičajen obrazac kod SPA – dohvat s poslužitelja putem Ajax poziva**

# Pogledajmo prvo server.js (iako je van fokusa)

ver3

server.js

```
const fs = require('fs');
let jsonRecipes;
fs.readFile('recipes.json',
    function (err, data) {
        var jsonData = data;
        jsonRecipes = JSON.parse(jsonData);
        console.log(jsonRecipes.length + " recipes parsed...");
    });

...
app.get('/recipes', function (req, res) {
    res.json(jsonRecipes);
});
```

ver3

# recipes.json

Preuzeto s:
https://github.com/fictivekin/openrecipes

recipes.js


```
[{
    "name": "Easter Leftover Sandwich",
    "ingredients": "12 whole Hard Boiled Eggs\n1/2 cup Mayonnaise\n3 Tablespoons Grainy Dijon Mustard\n Salt And Pepper, t
o taste\n Several Dashes Worcestershire Sauce\n Leftover Baked Ham, Sliced\n Kaiser Rolls Or Other Bread\n Extra Mayonnais
e And Dijon, For Spreading\n Swiss Cheese Or Other Cheese Slices\n Thinly Sliced Red Onion\n Avocado Slices\n Sliced Tomat
oes\n Lettuce, Spinach, Or Arugula",
    "url": "http://thepioneerwoman.com/cooking/2013/04/easter-leftover-sandwich/",
    "image": "http://static.thepioneerwoman.com/cooking/files/2013/03/leftoversandwich.jpg",
    "cookTime": "PT",
    "recipeYield": "8",
    "datePublished": "2013-04-01",
    "prepTime": "PT15M",
    "description": "Got leftover Easter eggs?    Got leftover Easter ham?    Got a hearty appetite?    Good! You've come t
o the right place!    I..."
},
{
    "name": "Pasta with Pesto Cream Sauce",
    "ingredients": "3/4 cups Fresh Basil Leaves\n1/2 cup Grated Parmesan Cheese\n3 Tablespoons Pine Nuts\n2 cloves Garlic,
 Peeled\n Salt And Pepper, to taste\n1/3 cup Extra Virgin Olive Oil\n1/2 cup Heavy Cream\n2 Tablespoons Butter\n1/4 cup Gr
ated Parmesan (additional)\n12 ounces, weight Pasta (cavitappi, Fusili, Etc.)\n2 whole Tomatoes, Diced",
    "url": "http://thepioneerwoman.com/cooking/2011/06/pasta-with-pesto-cream-sauce/",
    "image": "http://static.thepioneerwoman.com/cooking/files/2011/06/pesto.jpg",
    "cookTime": "PT10M",
    "recipeYield": "8",
    "datePublished": "2011-06-06",
    "prepTime": "PT6M",
    "description": "I finally have basil in my garden. Basil I can use. This is a huge development.    I had no basil dur
ing the winter. None. G..."
```

Recipes.js

ver3

```
export default class {
  async getHtml() {
      try {
          let response = await fetch('/recipes');
          if (response.ok) {
              let allRecipes = await response.json();
              let recipes = allRecipes.filter(x => Math.random()>0.75).reverse();
              return `<h1>Recipes (${recipes.length})</h1>
                    <div class="container-fluid p-2 d-flex flex-wrap">`
                   + recipes.map(rcp => `
               <div class="card mt-2 mr-2" style="width: 200px;">
               <img class="card-img-top" style="width: 190px;" src="${rcp.image}">
               <div class="card-body">
                 <h5 class="card-title">${rcp.name}</h5>
                 <p class="card-text">${rcp.description}</p>
                 <span class="badge badge-primary">
                     Cook/prep time: ${rcp.cookTime}/${rcp.prepTime}</span>
                 (...izbačen dio kod koji formatira...)
             </div>
               `).join('') + '</div>';
          } else { throw new Error("HTTP-Error: " + response.status); }
      } catch (error) {
          return `<h1>Home view (error occured)</h1>`;
      }
  }
}
```

Samo ~četvrtinu slučajno odabranih (ima ih 1042)

CookBook

localhost:8080/ver03//Recipes/Recipes

ver3

# My cookbook in the making ver3...

Recipes Calculator

## Recipes (274)

### Gougères

Gougères - I have these little cheese puffs in my freezer, ready to bake, nearly always. This version, a favorite, is made with whole wheat flour, sharp white cheddar cheese, fennel, and ale.

Cook/prep time: PT30M/PT10M
Yield:
Pusblished: 2011-12-17

Source

Ingredients

### Grilled Salt & Vinegar Potatoes

Salt & vinegar chip enthusiasts, these are for you. You take slabs of sliced potatoes, boil them in vinegar, then grill them to a crisp. Not for the faint of heart, or anyone with particularly sensitive taste buds ;)...

Cook/prep time: PT10M/PT35M
Yield:
Pusblished: 2010-07-11

Source

Ingredients

### Baked Sweet Potato Falafel Recipe

Baked Sweet Potato Falafel recipe from the Leon cookbook -made from mashed sweet potatoes, chickpea flour, spices, a nice amount of garlic and plenty of chopped cilantro.

Cook/prep time: / Yield:
Pusblished: 2009-05-17

Source

Ingredients

### Classic Cheese Fondue Recipe

A classic cheese fondue recipe. I've included dozens of my favorite things to dunk - don't limit yourself to just bread!

Cook/prep time: / Yield:
Pusblished: 2008-12-26

Source

Ingredients

### Caramelized Onion Dip Recipe

A grown-up remake of the onion dip I loved as a kid. This one features lots of deeply caramelized onions, sour cream and Greek yogurt.

Cook/prep time: /
Yield: Makes about 2 cups.
Pusblished: 2008-12-06

Source

Ingredients

### Hummus en Fuego Recipe

A beautiful, spicy hummus recipe made from pureed garbanzo beans, toasted walnuts, and spicy crushed red pepper oil finished with a few chopped olives and a bit of cilantro.

Cook/prep time: /
Yield: Makes roughly 2 1/2 cups..
Pusblished: 2008-10-08

Source

Ingredients

ver3

# Ver 3 osvrt

- ✓ Razložili smo aplikaciju na viewove
  - Pravi radni okviri će omogućiti i manje gradivne blokove - komponente
  - View se brine samo za sebe i stvaranje svog HTML-a
- ✓ (Proto)usmjeravanje, mijenjamo stranice na klijentu, kozmetički mijenjamo URL

- ✗ Loše:
  - ✗ Kako formiramo HTML – string concatenation, spaghetti code
- Treba nam:
  - Templating „engine”, elegantan sustav kako ćemo napraviti HTML

# Primjer: stvaranje HTML-a: React i Vue

```
class App extends React.Component {
    render() {
      const planets = ['Merkur', 'Venera', 'Zemlja'];
      return (
        <ol>
            {planets.length > 0 && planets.map(n => (
                <li key={"section-" + n}>Planet {n}</li>
            ))}
        </ol>
      );
    }
  }
```

JSX, or JavaScript XML, is an extension to the JavaScript language syntax

```
<ol>
  <li>Planet Merkur</li>
  <li>Planet Venera</li>
  <li>Planet Zemlja</li>
</ol>
```

```
<template>
  <div id="app">
    <ol>
      <li v-for="planet in planets">Planet {{ planet }}</li>
    </ol>
  </div>
</template>
<script>
export default {
  data() {
      return {
        planets: ['Merkur', 'Venera', 'Zemlja']
      }
    }
}; </script>
```

Ni jedno ni drugo nije validan JS, treba prevesti…

# V4 u sljedećem predavanju…

# Napredni razvoj programske potpore za web

**- predavanja -**

**2022./2023.**

## 8. Jednostranične web-aplikacije

## Vue.js

# Creative Commons

- slobodno smijete:
  - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
  - **prerađivati** djelo
- pod sljedećim uvjetima:
  - **imenovanje:** morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
  - **nekomercijalno:** ovo djelo ne smijete koristiti u komercijalne svrhe.
  - **dijeli pod istim uvjetima:** ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

*U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela.*
*Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.*
*Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.* *Tekst licence preuzet je s http://creativecommons.org/*

# µSPA – ver4

ver4

- Znamo:
  - Dinamički promijeniti sadržaj stranice
  - Presresti (i promijeniti) klik, ostvariti lokalno usmjeravanje i mijenjanje „stranica”
    - Koristimo vlastite oznake unutar HTML-a, npr. `data-link`
  - Razložiti kôd na module koji odgovaraju stranicama, stranice imaju jednostavne zadatke
  - Dohvatiti podatke s weba i formatirati HTML
- **Napravimo**:
  - **Model binding – povežimo HTML elemente s JS objektima (podatcima)**
  - **Automatske izračune ovisnih vrijednosti**
  - **Automatsko iscrtavanje (rendering)**

# Povezivanje podataka (UI data binding)

- Obrazac kod razvoja GUI aplikacija
- Povezivanje elemenata GUI-ja i domenskog modela

ver4

# Pogledajmo prvo klijentsku stranu – Calculator

Calculator.js

```
export default class  {
  constructor () {
    this.myData = {
        tempCelsius: 0,
        tempFahrenheit: 32
    };
  }
  async getHtml() {(...)}
  getData() { return this.myData; }
  onChange(key, value) {
    if (key === "tempCelsius") {
      this.myData.tempFahrenheit = parseFloat(value) * 9 / 5 + 32;
    } else if (key === "tempFahrenheit") {
      this.myData.tempCelsius = (parseFloat(value) - 32) * 5/ 9;
    }
  }
}
```

```
async getHtml() {
  return `
    <h1>Temperature calculator</h1>
    <form>...
      <input type="number"
         id="celsius" placeholder="C"
         uspa-bind="tempCelsius">
        ...
      <input type="number"
         id="fahrenheit" placeholder="F"
         uspa-bind="tempFahrenheit">
        ...</form>`;
}
```

Temperature calculator

Celsius

200

Fahrenheit

392

# „Digresija”: Proxy

ver4

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Proxy

ProxyExample.js

```
let obj = {  name: "Milivoj",  age: 45 };
let pObj = new Proxy(obj, {
  get: function (target, prop) {
    console.log("Netko želi znati ", prop);
    return target[prop];
  },
  set: function (obj, prop, value) {
    console.log("Netko postavlja ", prop, "na", value);
    if (prop === "age") {
      if (!Number.isInteger(value)) {
        throw new TypeError("Godine moraju biti cijeli broj!");
      }
    }
    obj[prop] = value;
    return true; // Indicate success
  },
});
console.log(obj.name, obj.age);
console.log(pObj.name, pObj.age);
try {
    pObj.age = "33a";
} catch (error) {
    console.error(error);
}
pObj.age = 33;
console.log(obj);
```

```
Milivoj 45
Netko želi znati  name
Netko želi znati  age
Milivoj 45
Netko postavlja  age na 33a
TypeError: Godine moraju biti cijeli broj!
    at Object.set (...digressions\ProxyExample.js:15:15)
    (...)
    at Object.<anonymous> (...\digressions\ProxyExample.js:26:14)
Netko postavlja  age na 33
{ name: 'Milivoj', age: 33 }
```

ver4

# Dodajmo i Observer obrazac

Observer.js

```
export default class Observer {
  constructor(dataObject, listener) {
    this.observersSet = [];
    if (listener) this.observersSet.push(listener);
    let self = this;
    this.proxyObject = new Proxy(dataObject, {
      set: (target, key, value, receiver) => {
        const result = Reflect.set(target, key, value, receiver);
        self.dataObjectClone = {...dataObject};  // used in isDirty()
        self.observersSet.forEach(observer => observer(key, value));
        return result;
      }
    });
  }
  isDirty() { // poor man's micro-optimization
    for (const key in this.dataObjectClone) {
      if (this.dataObjectClone[key] !== this.proxyObject[key]) { return true; }
    }
    return false;
  }
}
```

U našem slučaju, zainteresiran je:

`Calculator.onChange(key, value)`

ver4

# Konačno – povežimo sve

uspa.js

```
async setView(viewName) {
  let viewClass = this.currView.getViews()[viewName];
  this.currViewObject = new viewClass(); // should I cache it?
  await this.render();
  history.pushState(null, null, `${this.stubUrl}/${viewName}`);
  this.bindViewData(); }
bindViewData() {
  if (this.currViewObject.getData) {
    let currViewData = this.currViewObject.getData();
    this.currViewProxy = new Observer(currViewData,
                                      this.currViewObject.onChange.bind(this.currViewObject));
    document.querySelectorAll("[uspa-bind]").forEach((elem) => {
      let name = elem.getAttribute("uspa-bind");
      let proxy = this.currViewProxy.getProxy();
      elem.value = proxy[name];
      elem.onkeyup = () => {
        proxy[name] = elem.value;  // could use some metadata and do parsing, eg int, date, etc
        if (this.currViewProxy.isDirty()) {
          this.render(name);
        }
      };   });  } }
```

Izmjestili u posebnu metodu (sljedeći slajd)

ver4

# ...nastavak:

```
  async render(focusOnElementName) {  // silly rendering...
    document.querySelector("router-
view").innerHTML = await this.currViewObject.getHtml();
    this.bindViewData(); // ☹
    if (focusOnElementName) {
      document.querySelector(`[uspa-
bind="${focusOnElementName}"]`).focus();
    }
  }
```

# „Silly” rendering…

- Iscrtavanje na takav način bi bilo loše - kod svake promjene se:
  - ponovo iscrtava cijeli sadržaj
  - i ponovo se radi povezivanje svega!
- Različiti radni okviri koriste različite pristupa za optimiranje iscrtavanja, npr.:
  - React i Vue koriste tzv. Virtual DOM
  - Angular koristi „Incremental DOM” https://github.com/google/incremental-dom
  - Svelte ni jedno ni drugo, itd.
- S korisničke strane želi se postići da korisnik (developer) ne mora brinuti o (performansama) iscrtavanja, već da se bavi deklarativnim programiranjem i promjenama stanja

# Virtual DOM

- Nezahtjevna JS memorijska reprezentacija DOM-a

- Ideja:

1. promijeni model
2. (radni okvir ažurira VDOM)

radni okvir

Virtual DOM

body

div

input

input

div

p

...

3. diff

4. render

4. render

DOM

body

div

input

input

div

...

# Kraj primjera

- Vidjeli smo:
  - Routing
    - History API
  - Data-binding
    - Proxy, Observer
  - Rendering
    - Virtual DOM
  - (HTML) templates
  - „Components”

A sad idemo na pravi radni okvir…

# Vue.js

*a crash course…*

# Velika trojka: Angular, Vue, React

https://trends.google.com/trends/explore?q=%2Fg%2F11c6w0ddw9,%2Fg%2F11c0vmgx5d,%2Fm%2F012l1vxv

# Github stars

https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190

# Vue.js

- „nije važno” -> sva tri su dobra
- Wikipedia:
  - Vue.js (commonly referred to as Vue; pronounced /vjuː/, like "view") is an **open-source model–view–viewmodel front end** JavaScript **framework** for building **user interfaces** and **single-page applications**.
- Vue.js:
  - Vue is a **progressive framework** for building **user interfaces**.

- Evan You, 2014.:
  - *"I figured, what if I could just extract the part that I really liked about **Angular** and build something really lightweight.„*

https://web.archive.org/web/20170603052649/https://betweenthewires.org/2016/11/03/evan-you/

- *Vue 2 -> **Vue 3***

# 01-basics, interpolation, v-bind

# Primjer – Calculator

app.js

```
const Calculator = {
  data() {
    return {
      tempCelsius: 0,
      tempFahrenheit: 32,
      startedDateAt: new Date().toLocaleDateString("hr-HR"),
      startedTimeAt: new Date().toLocaleTimeString("hr-HR"),
    };
  },
};
const app = Vue.createApp(Calculator);
app.mount("#my-app");
```

HTML (template) je zasad u index.html (sljedeći slajd)

# Primjer – Calculator

index.html

```
<body>
  <div id="my-app">
    <h3>Temperature calculator, the time is: {{ startedDateAt }} {{ startedTimeAt }} </h3>
    <form>
      <div class="form-group row">
        <label for="celsius" class="col-sm-2 col-form-label">Celsius</label>
        <div class="col-sm-10">
          <input type="number" class="form-control" placeholder="C"
            v-bind:value="tempCelsius">
        </div>
      </div>
      <div class="form-group row">
        <label for="fahrenheit" class="col-sm-2 col-form-label"
        <div class="col-sm-10">
          <input type="number" class="form-control" placeholder=
           :value="tempFahrenheit">
        </div>
      </div>
    </form>
  </div>
</body>
```

*interpolation: {{ js }}*

*v-bind:* jednosmjerno

*v-bind:* skraćena sintaksa

Temperature calculator, the time is: 18. 05. 2021. 16:28:59

Celsius

0

Fahrenheit

32

# 02-basics, events, methods, this

# Dodajmo interakciju – prvo dodajem methods

app.js

```
const Calculator = {
  data() {
    return {
      tempCelsius: 0,
      tempFahrenheit: 32, (...)
    };
  },
  methods: {
    c2f() {
      this.tempCelsius = this.toFloat(this.$refs.tempCelsius.value);
      this.tempFahrenheit = Math.round(this.tempCelsius  * 9 / 5 + 32);
    },
    f2c(event) {
      event.preventDefault();
      this.tempFahrenheit = this.toFloat(this.$refs.tempFahrenheit.value);
      this.tempCelsius = Math.round((this.tempFahrenheit  32) * 5/ 9);
    },
    toFloat(value) {  // suvišno, samo da se vidi this.method
      return parseFloat(value);
    }
  }
};
```

Pored data, sad imamo i methods

Očigledno, svojstva podatkovnog objekta i funkcije u methods bivaju mapirane na this *(Proxy)*

Pozivat ćemo ih onclick; ako nas zanima Vue će nam poslati i event object.

Ugrađeni $refs objekt će sadržavati HTML elemente koje označimo (sljedeći slajd)

# Dodajmo interakciju (pretvorbu)

index.html

```
<form>
  <div class="form-group row">
    <label for="celsius" class="col-sm-4 col-form-label">Celsius</label>
    <div class="col-4">
      <input type="number" class="form-control" placeholder="C"
        v-bind:value="tempCelsius" ref="tempCelsius">
    </div>
    <div class="col-2">
      <button v-on:click="c2f">→F</button>
    </div>
  </div>
  <div class="form-group row">
    <label for="fahrenheit" class="col-4 col-form-label" >Fa
    <div class="col-4">
      <input type="number" class="form-control" placeholder=
        :value="tempFahrenheit" ref="tempFahrenheit">
    </div>
    <div class="col-2">
      <button @click.prevent="f2c">→C</button>
    </div>
  </div>
</form>
```

dodajemo u $refs, ime je proizvoljno

v-on:event: method ili method(...)

*v-on:* skraćena sintaksa @

event modifier – otkazuje submit

# 03-basics, v-model, lifecycle

# Vue3 Lifecycle

Dva API-ja, zato postoje parovi *Lifecycle Hook* funkcija.

https://learnvue.co/2020/12/how-to-use-lifecycle-hooks-in-vue3/
Pogledati i:
https://vuejs.org/guide/essentials/lifecycle.html#lifecycle-diagram

# v-model: povezujemo UI i varijablu modela (i uklanjamo gumbe)

index.html

```
<h3>Temperature calculator, the time is: {{ startedDateAt }} {{ startedTimeAt }
} </h
<form
    <
    <label      elsius" class="col-sm-4 col-form-label">Celsiu
    <div c      "col-4">
        <in  ut type="number" class="form-control" placeholder="C"
        v-model="tempCelsius" @keyup="c2f">
    </div>
    </div>
    <div class="form-group row">
    <label for="fahrenheit" class="col-4 col-form-label" >Fahrenheit</label>
    <div class="col-4">
        <input type="number" class="form-control" placeholder="F"
        v-model.number="tempFahrenheit" @keyup.enter="f2c">
    </div>
    </div>
</form>
```

***Two-way binding***: v-model

Mijenjat ćemo svake sekunde! Što je s iscrtavanjem?

Kod svake promjene izračunavamo F

***Binding modifier***: vue će pretvarati string u number

***keyup modifier***: samo ako je enter

# Lifecycle event - mounted

app.js

```
methods: {
  c2f() {
    this.tempFahrenheit = Math.round((this.tempCelsius * 9) / 5 + 32);
  },
  f2c() {
    console.log(typeof this.tempFahrenheit);
    this.tempCelsius = Math.round(((this.tempFahrenheit - 32) * 5) / 9);
  }
},
mounted() {
  window.setInterval(() => {
    this.startedTimeAt = new Date().toLocaleTimeString("hr-HR");
  }, 1000);
}
```

Primijetiti da više ne moramo postavljati C, lokalna varijabla `this.tempCelsius` je automatski ažurna

Zbog binding modifera ovo će biti number

Lifecycle event: https://v3.vuejs.org/guide/instance.html#lifecycle-diagram

Svaku sekundu ažuriramo vrijednost jedne varijable modela i ne brinemo za iscrtavanje!

# 04-basics: v-for, v-if, :class, template

# Dodajmo novu varijablu i dvije funkcije

app.js

```
...
data() {
  return {
    (...)
    logItems: []
  };
},
methods: {
  c2f() { ... },
  f2c() { ... },
  logItemClass(item) {
    return  (item.C > 200) ? "hot" : "";
  },
  logTemp() {
    this.logItems.push({
      C: this.tempCelsius,
      F: this.tempFahrenheit
    })
  }
},
```

Dodajemo polje

Dodajemo trenutne vrijednosti temperatura u polje

styles.css

```
...
.hot::after {
    content: " \1F525";
    color: red;
    font-weight: bold;
}
```

# Preselili HTML iz index.html -> app.js

app.js

```
const Calculator = {
  template: `<h3>Temperature calculator, the time is: {{starte
  <form>
      (...)
    <div class="form-group row">
        <div class="col-12">
            <button type="button" @click.prevent="logTemp">Log it</button>
        </div>
    </div>
    <div v-if="logItems.length > 0">
      <h2>Logged temperatures:</h2>
      <div class="form-group row">
          <ul>
            <li v-for="item in logItems" :class="logItemClass(item)">
                {{ item.C }}°C = {{ item.F }}°F
            </li>
          </ul>
      </div>
    </div>
</form> `,
  data() {...
```

index.html

```
<body>
    <div id="my-app">
    </div>
</body>
```

Izgubili smo *syntax coloring*, vidjet ćemo kasnije bolji način za definirati template unutar „komponente”

Ako je polje prazno, nema cijelog div bloka

Iteriramo po elementima polja, postoji i sintaksa za dobiti indekse

Bind klase (class), metoda će vratiti ime. Nije u koliziji s eventualno ručno postavljenim klasama.

# Novi primjer – Hanojski tornjevi

## https://en.wikipedia.org/wiki/Tower_of_Hanoi

hanoi-01

# Inicijalne postavke – HTML, CSS

index.html

```
...
<div id="app">
    <div class="board">
        <div class="rod">
        </div>
        <div class="rod">
        </div>
        <div class="rod">
        </div>
    </div>
</div>
...
```

hanoi.css

```
div.board {
  margin: auto;
  width: 1000px;
  border-radius: 5px;
  box-shadow: 2px 2px 2px 1px rgba(0, 0, 0, 0.2);
  display: flex;
  justify-content: space-around;
}
div.rod {
  width: 300px;
  height: 500px;
  background-image: url("./stick.png");
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
  display: flex;
  align-items: center;
  flex-direction: column;
  justify-content: flex-end;
  padding-bottom: 107px;
}
div.disk {
  height: 40px;
  border-radius: 5px;
  border: 2px solid black;
}
```

# v-for, style – iscrtavamo N diskova

hanoi-02

index.html

```
...
<div class="board">
  <div class="rod">
    <div v-for="disk in positionA"
      class="disk"
      @click="onSelectFrom(disk)"
      :style="rodStyle(disk)">
    </div>
  </div>
  <div class="rod">
    <div v-for="disk in positionB"
      class="disk"
      @click="onSelectFrom(disk)"
      :style="rodStyle(disk)">
    </div>
  </div>
  <div class="rod">
    <div v-for="disk in positionC"
      class="disk"
      @click="onSelectFrom(disk)"
      :style="rodStyle(disk)">
    </div>
  </div>
</div>
...
```

hanoi.js

Nova sintaksa

```
var app = new Vue({
  el: '#app',
  data: {
    diskNumber: 3,
    positionA: [1, 2, 3],
    positionB: [],
    positionC: [],
  },
  methods: {
    rodStyle(diskSize){
      return {
        width: (diskSize / this.diskNumber * 60 + 20) + '%',
        backgroundColor: this.getRodColor(diskSize)
      }
    },
    getRodColor(diskSize){
      const colors = ['green', 'red', 'blue',
          'orange', 'yellow', 'purple'];
      return colors[diskSize % colors.length];
    }
  }
})
```

# Prije nego nastavimo…

- Instalirajte Vue devtools:
  - https://chrome.google.com/webstore/search/vue%20devtools
- https://vueschool.io/lessons/using-vue-dev-tools-with-vuejs-3

- Ručno mijenjajte position* polja

# Napredni razvoj programske potpore za web

**- predavanja -**

**2022./2023.**

## 9. Jednostranične web-aplikacije

## Vue.js

# Creative Commons

- slobodno smijete:
  - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
  - **prerađivati** djelo
- pod sljedećim uvjetima:
  - **imenovanje:** morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
  - **nekomercijalno:** ovo djelo ne smijete koristiti u komercijalne svrhe.
  - **dijeli pod istim uvjetima:** ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

*U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela.*
*Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.*
*Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.* *Tekst licence preuzet je s http://creativecommons.org/*

# Omogućimo vizualno isticanje odabranog diska, prebacivanje diskova (uz pravila), te „dnevnik”

| | |
|---|---|
| 21. 05. 2021. 11:11:56 | No disk may be placed on top of a disk that is smaller than it. |
| 21. 05. 2021. 11:04:48 | Selected disk: 2 at rod A |
| 21. 05. 2021. 11:04:47 | Moved disk 1 from A to B |
| 21. 05. 2021. 11:04:45 | Selected disk: 1 at rod A |

**Congrats, game over in 7 moves!** **That is optimal.**

| | |
|---|---|
| 21. 05. 2021. 11:14:00 | Moved disk 1 from A to B |
| 21. 05. 2021. 11:13:59 | Selected disk: 1 at rod A |
| 21. 05. 2021. 11:13:58 | Moved disk 2 from C to B |
| 21. 05. 2021. 11:13:57 | Selected disk: 2 at rod C |
| 21. 05. 2021. 11:13:56 | Moved disk 1 from C to A |
| 21. 05. 2021. 11:13:55 | Selected disk: 1 at rod C |
| 21. 05. 2021. 11:13:54 | Moved disk 3 from A to B |
| 21. 05. 2021. 11:13:53 | Selected disk: 3 at rod A |
| 21. 05. 2021. 11:13:52 | Moved disk 1 from B to C |
| 21. 05. 2021. 11:13:51 | Selected disk: 1 at rod B |

# Reactivity

index.html

```
...
<div v-if="isGameOver" class="board card">
    <h1>Congrats, game over in {{ moves }} moves!</h1>
    <h2>That is {{ moves !== (Math.pow(2, diskNumber)-1) ? "SUB" : "" }}optimal.</h2>
</div>
<div v-else class="board card">
  <div class="rod" @click="onSelectTo('A')">
    <div v-for="disk in positionA"
       class="disk"
       :class="{selectedDisk : disk === selectedDisk, inactiveDisk: selectedDisk != null && disk !== selectedDisk}"
       @click="onSelectFrom(disk, 'A')"
       :style="rodStyle(disk)">
    </div>
  </div>
Analogno za druga dva...
```

v-if je igra gotova, ispišemo je li optimalan broj poteza, v-else prikazujemo igru (board)

(uvjetno) postavljamo klase selectedDisk i inactiveDisk

hanoi.js

```
var app = new Vue({
  el: '#app',
  data: { diskNumber: 3,
    positionA: [1, 2, 3],   positionB: [],    positionC: [],
    selectedDisk: null,     selectedRod: null,
    logMessages: [],
    moves: 0
  },
  computed: {
    isGameOver: function() {
      return this.positionB.length === this.diskNumber
        || this.positionC.length === this.diskNumber
    }
  }, ... nastavak na sljedećem slajdu...
```

computed property – kada imamo iole složeniju logiku u templateima (koja se temelji na modelu) možemo ju premjestiti u computed property. Npr. izračun optimalnog broja poteza gore ili klase se također mogao „izmjestiti”

# Reactivity: odabir i spuštanje diska

hanoi.js

```
methods: { ...
  onSelectFrom(disk, rod) {
      if (
      (this.positionA[0] === disk)
      || (this.positionB[0] === disk)
      || (this.positionC[0] === disk)
      ) {
      this.selectedDisk = disk;
      this.selectedRod = rod;
      this.log("Selected disk: "
     + this.selectedDisk + " at rod "
      + this.selectedRod);
      } else {
          this.log("You can only select the
topmost disk.");
      }
  },
  log(msg) {
     this.logMessages.unshift({
     ts: new Date().toLocaleString("hr-HR"),
     text: msg,
   });
  },
...
```

hanoi.js

```
onSelectTo(rod) {
    if (this.selectedDisk
         && rod !== this.selectedRod) {
    let positionABC = [this.positionA, this.positionB,
                       this.positionC];
    let idxTo = rod.charCodeAt(0) - "A".charCodeAt(0);
    let rodToArray = positionABC[idxTo];
    let idxFrom = this.selectedRod.charCodeAt(0)-"A".charCodeAt(0);
    let rodFromArray = positionABC[idxFrom];

    if (rodToArray.length && rodToArray[0] < this.selectedDisk) {
        this.log(
   "No disk may be placed on top of a disk that is smaller than it."
        );
    } else {
        rodFromArray.shift();
        rodToArray.unshift(this.selectedDisk);
        this.log("Moved disk " + this.selectedDisk + " from " +
            this.selectedRod + " to " +  rod
        );
        this.selectedRod = null;
        this.selectedDisk = null;
        this.moves++;
    }
  }
}
```

# Ispis dnevnika i CSS

index.html

```
...
<div class="log card">
    <div v-for="msg in logMessages" >
      <span class="log--time">
        {{ msg.ts }}
      </span>
      <span class="log--message">
         {{ msg.text }}
      </span>
    </div>
</div>
```

| | |
|---|---|
| 21. 05. 2021. 11:14:00 | Moved disk 1 from A to B |
| 21. 05. 2021. 11:13:59 | Selected disk: 1 at rod A |
| 21. 05. 2021. 11:13:58 | Moved disk 2 from C to B |
| 21. 05. 2021. 11:13:57 | Selected disk: 2 at rod C |
| 21. 05. 2021. 11:13:56 | Moved disk 1 from C to A |
| 21. 05. 2021. 11:13:55 | Selected disk: 1 at rod C |
| 21. 05. 2021. 11:13:54 | Moved disk 3 from A to B |
| 21. 05. 2021. 11:13:53 | Selected disk: 3 at rod A |
| 21. 05. 2021. 11:13:52 | Moved disk 1 from B to C |
| 21. 05. 2021. 11:13:51 | Selected disk: 1 at rod B |

hanoi.css

```
.selectedDisk {
  margin: 5px;
  box-shadow: 0px 0px 5px 5px rgba(0, 255, 0, 0.5),
  0px 0px 5px 5px inset rgba(255, 238, 0, 0.5);
}
.inactiveDisk {
  box-shadow: 0px 0px 20px 10px inset rgba(0, 0, 0, 0.7);
}
div.log {
  margin-top: 20px;
  max-height: 200px;
  overflow: auto;
  min-height: 100px;
  font-family: Verdana, Geneva, Tahoma, sans-serif;
}
.log--time {
  font-size: 0.9em;
  background-color: honeydew;
  padding: 0 5px 0 5px;
  border: 1px solid gray;
  border-radius: 2px;
}
```

# Par napomena uz *computed property*

- Sličnost s `methods`
  - Kada koristiti jedno, kada drugo?
    - **Computed properties** - kada samo mijenjamo prezentaciju, ali ne i podatke
      - Štoviše - **paziti da ne mijenjamo podatke** unutra *computed properties* - nezgodni bugovi!
        pogledajte: https://vueschool.io/lessons/computed-properties-in-vue-3?friend=vuejs
    - **Methods** - kada mijenjamo podatke
  - Bitna razlika - CP su **keširani**!
    - Ponovo se izračunavaju samo ako se promijeni neka **reaktivna** varijabla na temelju koje se izračunava
    - Npr. nikad se neće osvježiti:
    - (moguće isključiti cache)

```
computed: {
  now() {
    return Date.now()
  }
}
```

- Postoji i Computed setter

# Omogućimo novu igru s različitim brojem diskova, izračunajmo score i zapamtimo ga kako se ne bi izgubio nakon osvježavanja stranice

hanoi-04

# Dodajemo highScore, učitavamo, snimamo…

hanoi-04

index.html

```
<div class="board card">
  <div class="reset-game">
  <select ref="disksNumber"
      class="form-control">
      Number of disks:
      <option value="2">2 disks</option>
      <option value="3">3 disks</option>
      <option value="4">4 disks</option>
      <option value="5">5 disks</option>
  </select>
  <button @click="resetGame"
      class="form-control">Reset game
  </button>
  </div>
    <div class="high-score"
     :class="{ newHighScore: isNewHighScore }">
    Current score: {{score}}
    High score: {{highScore}}
    </div>
</div>
```

hanoi.js

```
computed: { …
  score: function () {
    return this.diskNumber * 10 -
       11 * (this.moves - this.optimalMoves());
  },
  isNewHighScore: function () {
    return this.isGameOver
       && (this.score > this.highScore);
    }
},  methods: { …
onSelectTo(rod) {
    …
    this.moves++;
    if (this.isNewHighScore) {
      this.highScore = this.score;
      localStorage.setItem('highScore', this.highScore);
      this.log("** CONGRATS!!! NEW HIGH SCORE!! **");
    }
  }
},
beforeMount() {
  let highScore = localStorage.getItem('highScore');
  if (highScore != null) {
    this.highScore = parseInt(highScore);
} …
```

Lifecycle event

# Vue.js

*a crash course…*

*routing, components, forms, Pinia*

# Inicijalizacija Vue aplikacije

https://vuejs.org/guide/quick-start.html#creating-a-vue-application

```
λ npm init vue@latest
Need to install the following packages:
  create-vue@latest
Ok to proceed? (y) y

Vue.js - The Progressive JavaScript Framework

√ Project name: ... fer-demo-spa
√ Add TypeScript? ... No / Yes
√ Add JSX Support? ... No / Yes
√ Add Vue Router for Single Page Application development? ... No / Yes
√ Add Pinia for state management? ... No / Yes
√ Add Vitest for Unit Testing? ... No / Yes
√ Add an End-to-End Testing Solution? » No
√ Add ESLint for code quality? ... No / Yes
√ Add Prettier for code formatting? ... No / Yes

Scaffolding project in C:\web2\fer-demo-spa...

Done. Now run:

  cd fer-demo-spa
  npm install
  npm run lint
  npm run dev
```

# Struktura projekta

src > main.js > ...

```
1  import { createApp } from "vue";
2  import { createPinia } from "pinia";
3
4  import App from "./App.vue";
5  import router from "./router";
6
7  import "./assets/main.css";
8
9  const app = createApp(App);
10
11 app.use(createPinia());
12 app.use(router);
13
14 app.mount("#app");
```

index.html > ...

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8">
5      <link rel="icon" href="/favicon.ico">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Vite App</title>
8    </head>
9    <body>
10     <div id="app"></div>
11     <script type="module" src="/src/main.js"></script>
12   </body>
13 </html>
```

# Single-File Components (.vue datoteke)

```
<template>
  (ovdje pišemo HTML)
</template>


<script>
  (ovdje pišemo JS)
</script>


<style scoped>
  (ovdje pišemo CSS)
</style>
```

*https://vuejs.org/guide/scaling-up/sfc.html*

- Imamo tri sekcije:
  - template
  - script
  - style
- Vite automatski prevodi u JS
  - https://vitejs.dev/guide/why.html
- Koristi se:
  - View
  - Component

# View vs Component

- Primjećujemo dva koncepta u ustroju projekta (i dva različita direktorija):
  - **Views** (`src/views`)
  - **Components** (`src/components`)
- Na tehničkoj razini su to iste stvari – **komponente**
  - Vrijedi sve s prethodnog slajda
- Ali semantički:
  - **Views** predstavljaju **stranice** naše **više-stranične** aplikacije i tipično se referenciraju iz usmjerivača (`router`, uskoro)
  - **Components** predstavljaju (ponovo upotrebljive) komponente koje se koriste na N stranica
- Postoje **različite konvencije** kako ih nazivati i gdje ih pohranjivati (npr. *pages, views*, itd.)

# Izmijenimo generirani projekt u začetak naše Cookbook aplikacije

src > views > RecipesView.vue > ...

```
1  <template>
2    <h1>My cookbook in the making... Vue ver01</h1>
3  </template>
```

127.0.0.1:5173

Recipes | Calculator

My cookbook in the making... Vue ver01

src > views > CalculatorView.vue > ...

```
1  <template>
2    <h1>Convert C to F and vv.</h1>
3  </template>
```

127.0.0.1:5173/calculator

Recipes | Calculator

Convert C to F and vv.

Ako koristite **VS Code**, instalirati **Volar** ~~**Vetur**~~ za *syntax highlighting* i sl.

# Konfiguracija routera

ver01

main.js

```
import router from "./router";
...
app.use(router);
...
```

App.vue

```
<script setup>
import { RouterLink, RouterView } from "vue-router";
</script>
<template>
  <header>
    <nav class="d-flex justify-content-center bg-light border border-bottom border-dark p-3 w-100">
    <RouterLink to="/">Recipes</RouterLink>
      |
    <RouterLink to="/calculator">Calculator</RouterLink>
    </nav>
  </header>
  <RouterView />
</template>
<style scoped>
nav a {
  margin-left: 1rem;  margin-right: 1rem;
}
</style>
```

router/index.js

```
import { createRouter, createWebHistory } from "vue-router";
import RecipesView from "../views/RecipesView.vue";
import CalculatorView from "../views/calculatorView.vue";
const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: "/",
      name: "recipes",
      component: RecipesView,
    },
    {
      path: "/calculator",
      name: "calculator",
      component: CalculatorView,
     },
  ],
});
export default router;
```

ver02

# Dynamic routing, catch-all

router/index.js

```
...
const routes = [ // postoji i alias:"/"
  { path: "/", // mogli smo i redirect: "/recipes
    component: Recipes,
  },
  {
    path: "/recipes/:id?",
    component: Recipes,
  },
  { path: "/calculator",
    component: Calculator,
  },
  { path: "/:catchAll(.*)",
    name: "NotFound",
    component: NotFound,
  },
];
...
```

Opcionalni (zbog upitnika) parametar, poslije dostupan kao `$route.params.id`

NotFound.vue

```
<template>
  <h1>Sorry, {{ $route.params.catchAll }}
    is not found.</h1>
  Try <router-link to="/">home</router-link>
</template>
```

catchAll je proizvoljno ime varijable, u zagradi je regex

# Recipes.vue 1/2

ver02

Recipes.vue

```
<template>
  <div v-if="$route.params.id">
    You are looking for recipe id {{ $route.params.id }}
    <br>Try something <router-link :to="'/recipes/' + somethingElse">
      else e.g. {{ somethingElse }} ...
    </router-link>
  </div>
  <div v-else>All recipes</div>
  <ul>
    <li v-for="msg in routeChanges"
      :key="msg">{{ msg }}</li>
  </ul>
</template>
<script>
export default {
  data() {
    return {
      routeChanges: [],
      somethingElse: Math.floor(Math.random() * 1000)
    };
  }, ...
```

Primijetiti dvotočku – izraz pod navodnicima se onda tumači kao js expression

# Recipes.vue 2/2: watchers, scoped style

ver02

Recipes.vue

```
...
watch: {
    $route(to, from) {
      this.routeChanges.push(`Route changed: ${from.path} -> ${to.path} `);
      this.somethingElse = Math.floor(Math.random() * 1000);
    },
  },
};
</script>

<style scoped>
* {
  font-size: 32px;
  color: navy;
}
</style>
```

watch opcija – moguće je pratiti promjene neke varijable! Inače ne bi detektirali promjenu rute, jer Vue čuva komponentu u memoriji

Primijetiti da je stil primijenjen samo na view, npr. naslov je i dalje crne boje, zašto?

DM1

# Što će se dogoditi ako ručno upišemo npr. /recipes/101 ?

- Uočiti razliku:

Ako ga ne konfiguriramo, WS će vjerojatno vratiti 404.

*Npr. probajte npm run build i onda dist folder pogledati preko LiveServera*

GET /recipes/101

**Web server**

NotFound.vue

```
<template>
  ...
  Try <router-link to="/recipes/101">
        recipe 101
  </router-link>
</template>
```

## Slide 20

**DM1** ovo je komentar nastavno na zadatak za vježbu? zato sto je slajd izmedju zadatka za vjezbu i komponente

Danijel Mlinarić; 15.9.2021.

# Zadatak za vježbu

Pogledajte *programmatic navigation*, te dodajte na `Recipes.vue` *button* koji će korisnika prebaciti na **slučajni recept**

# Komponente

# Komponente

- Kada:
  - Imamo dijelove stranice/markupa koji se ponavljaju
  - Želimo razložiti problem/aplikaciju u manje probleme/aplikacije
- Komponenta je poput mini-aplikacije koja se onda pridijeli aplikaciji
  - Pridjeljujemo im oznaku (*tag*), tipično dvije riječi odvojene crticom kako bi izbjegli preklapanje s HTML oznakama
- Uobičajeno se aplikaciju ustroji u stablo ugniježđenih komponenti,
  npr. zaglavlje, tijelo, podnožje, popup/dialog, …

https://v3.vuejs.org/guide/component-basics.html

# Obrasci komunikacije

- **Roditelj -> dijete**
  - Tipično se komponentama predaju vrijednosti
- Dijete -> roditelj
  - Rjeđe, ali ponekad nužno
  - Npr. React ima „*one-way data flow parent → child*”, ali…

# Kakva može biti komponenta?

- Tri obilježja:
  1. **Što prima?**
  2. **Ima li vlastito stanje?**
  3. **Surađuje li s drugima?**
     - Po pitanju podataka
     - Omoguće li gniježđenje, proširivanje
- Različiti radni okviri više ili manje razrađuju nomenklaturu s obzirom na navedeno, npr.
  - React ima *class* i *functional* components
  - Angular: *smart i dumb/presentational/pure* komponente (samo primaju podatke i iscrtavaju)

# Pretvorimo karticu recepta i kalkulator u komponente

ver03

- Prvo jednostavniju – Calculator
  - Vlastito stanje, ne prima parametre, ne surađuje

main.js

```
import { createApp } from "vue";
import App from "./App.vue";
import router from "./router";
import RecipeCard from './components/RecipeCard.vue';
import TempConverter from './components/TempConverter.vue';

const app = createApp(App);
app.use(router);
app.component('recipe-card', RecipeCard);
app.component('temp-converter', TempConverter);
app.mount("#app");
```

Konvencija: xxx-yyy

Prijavljujemo ih „globalno” čime postaju dostupne u cijeloj aplikaciji. (sljedeći slajd)

Views/CalculatorView.vue

```
<template>
  <div>
    <h2>Calculator</h2>
    <div class="container-fluid p-2 d-flex justify-content-center">
      <temp-converter></temp-converter>
    </div>
  </div>
</template>
```

# Vue komponente - komentar

- Da bi koristili komponente, moramo ih registrirati
- Dvije mogućnosti:
  1. Globalno registrirati komponentu -> vidljiva svugdje
     - Nedostatci:
       - Onemogućen *treeshaking*
       - Teži uvid u ovisnost/povezanost komponenti
  2. Lokalno registrirati komponentu
     - Vidljivo samo u toj komponenti, ne i njenoj djeci

```js
import { createApp } from 'vue'

const app = createApp({})

app.component(
  // the registered name
  'MyComponent',
  // the implementation
  {
    /* ... */
  }
)
```

If using SFCs, you will be registering the imported `.vue` files:

```js
import MyComponent from './App.vue'

app.component('MyComponent', MyComponent)
```

# TempConverter.vue

ver03

- Prvo jednostavniju – Calculator
  - Ima vlastito stanje, ne prima parametre, ne surađuje

Svaki dio se može izostaviti, npr. ovdje nema style

struktura:

```
<template>
   ...
</template>
<script >
   ...
</script >
<style>
   ...
</style>
```

components/TempConverter.vue

```
<template>
  <div><h3>Temperature converter</h3>
    <form>
      <div><label>Celsius</label>
        <div><input
            v-model.number="tempCelsius"
            @keyup="c2f"/>
        </div>
      </div>
      <div><label>Fahrenheit</label>
        <div><input
            v-model.number="tempFahrenheit"
            @keyup="f2c"
          />
        </div></div>
    </form></div>
</template>
```

Izbačeni svi stilovi i sitnice koje nisu u fokusu, pogledati src

components/TempConverter

```
<script>
export default {
  data() {
    return {
      tempCelsius: 0,
      tempFahrenheit: 32,
    };
  },
  methods: {
    c2f() {
      this.tempFahrenheit = Math.round((this.tempCelsius
            * 9) / 5 + 32);
    },
    f2c() {
      this.tempCelsius = Math.round(((this.tempFahrenheit
            - 32) * 5) / 9);
    },
  },
};
</script>
```

# Domaća zadaća

- Primijenite komponente, dodajte joj „Log it” gumb koji smo već vidjeli
- Listu temperatura možete prikazati kao tooltip ili sl.

- Time će naša komponente proširiti svoje lokalno stanje (pored temperatura C i F)
- Stavite nekoliko istih komponenti na npr. tu istu stranicu
- Što se događa kada otiđete na stranicu recepata i vratite se?
  - Pogledajte i `keep-alive`
    - Može se keširati i komponenta i stranica, pazite https://stackoverflow.com/questions/40898440/vue-2-keep-alive-not-working-with-router-view-and-key

# Vue props

- Pogledajmo: https://v3.vuejs.org/guide/component-props.html
- ***Static***:

```
<blog-post title="My journey with Vue"></blog-post>
```

- ***Dynamic:***

```
<!-- Dynamically assign the value of a variable -->
<blog-post :title="post.title"></blog-post>

<!-- Dynamically assign the value of a complex expression -->
<blog-post :title="post.title + ' by ' + post.author.name"></blog-post>
```

- Mogu se predavati različiti tipovi: number, boolean, array,…
- ***Object***:
- Itd. pogledati dokumentaciju

```
<!-- Even though the object is static, we need v-bind to tell Vue that -->
<!-- this is a JavaScript expression rather than a string.             -->
<blog-post
  :author="{
    name: 'Veronica',
    company: 'Veridian Dynamics'
  }"
></blog-post>

<!-- Dynamically assign to the value of a variable. -->
<blog-post :author="post.author"></blog-post>
```

# Citat iz dokumentacije:

## # One-Way Data Flow

All props form a **one-way-down binding** between the child property and the parent one: when the parent property updates, it will flow down to the child, but not the other way around. This prevents child components from accidentally mutating the parent's state, which can make your app's data flow harder to understand.

In addition, every time the parent component is updated, all props in the child component will be refreshed with the latest value. This means you should **not** attempt to mutate a prop inside a child component. If you do, Vue will warn you in the console.

There are usually two cases where it's tempting to mutate a prop:

1. **The prop is used to pass in an initial value; the child component wants to use it as a local data property afterwards.** In this case, it's best to define a local data property that uses the prop as its initial value:

```js
props: ['initialCounter'],
data() {
  return {
    counter: this.initialCounter
  }
}
```

2. **The prop is passed in as a raw value that needs to be transformed.** In this case, it's best to define a computed property using the prop's value:

```js
props: ['size'],
computed: {
  normalizedSize() {
    return this.size.trim().toLowerCase()
  }
}
```

**Note**

Note that objects and arrays in JavaScript are passed by reference, so if the prop is an array or object, mutating the object or array itself inside the child component **will** affect parent state.

# RecipeCard.vue 1/2 (script, style)

ver03

- Nema stanje, **prima parametre**, ne surađuje

components/RecipeCard.vue

```
<script>
export default {
  props: [
    "id", "image", "name", "description", "cookTime",
    "prepTime", "recipeYield", "datePublished", "url",
    "ingredients"
  ],
  computed: {
    idUrl() {
      return '/recipes' + this.id;
    }
  }
};
</script>
<style scoped>
  div.card-body .badge {
      white-space: pre-wrap;
  }
</style>
```

Template na sljedećem slajdu, uočiti props

Scoped – lokalni stil, utječe samo na ovu komponentu!

views/RecipesView.vue

```
<template>
  <div v-if="selectedRecipe">
    Recipe #{{ id }}
    <recipe-card :key="selectedRecipe.id"
        v-bind="selectedRecipe"
        :zoom="1"
      ></recipe-card>
  </div>
  <div v-else>
    <h2>All recipes ({{allRecipes.length}})</h2>
    <div>
      <recipe-card v-for="recipe in allRecipes"
        :key="recipe.id"
        v-bind="recipe"
      ></recipe-card>
    </div>
  </div>
</template>
```

*Static property*

*„If you want to pass all the properties of an object as props, you can use v-bind without an argument (v-bind instead of :prop-name). ”*

# RecipeCard.vue 2/2 (template)

ver03

components/RecipeCard.vue

```
<template>
  <div class="card mt-2 mr-2" :style="{ width: 230 * (zoom ? 3 : 1) + 'px' }">
    <router-link :to="'/recipes/' + id" class="mr-3">
      <img class="card-img-top w-100" :src="image" />
    </router-link>
    <div class="card-body">
      <h5 class="card-title">{{ name }}</h5>
      <p class="card-text">{{ description }}</p>
      <ul class="list-group">
        <li class="list-group-item">
          Cook/prep time: {{ cookTime }}/{{ prepTime }}
        </li>
        <li class="list-group-item">Yield: {{ recipeYield }}</li>
        <li class="list-group-item">Published: {{ datePublished }}</li>
        <li class="list-group-item"><a :href="url" target="_blank" class="card-link">{{ url.substring(0, 20) }}...</a></li>
      </ul>
    </div>
    <details v-if="ingredients">
      <summary><h3>Ingredients</h3></summary>
      <ol class="list-group">
        <li
          v-for="ingredient in ingredients"
          :key="ingredient"
          class="list-group-item"
        >
          {{ ingredient }}
        </li>
      </ol>
    </details>
  </div>
</template>
```

Ako je zoom postavljen, koristimo širu karticu

Nema ničeg posebno novog, jednostavno koristimo props kao normalne varijable kod definiranja obrasca odnosno iscrtavanja

# ver03

127.0.0.1:5173
70%
My cookbook (ver03)
Recipes | Calculator
All recipes (96)
Baked Polenta Fries Recipe
Great party finger food. A thick slab of polenta is sliced into the shape of a french fry and baked off.
Cook/prep time: /
Yield: Makes 2 dozen wide-cut fries.
Published: 2006-08-13
http://www.101cookbo...
Ingredients
Heirloom Tomato Tart in a Parmesan Crust
A knock-out tomato tart recipe - it highlights the flavor and vibrancy of the tomatoes while remaining mush-free.
Cook/prep time: /
Yield:
Published: 2005-08-05
http://www.101cookbo...
Ingredients
Thinnest Oatmeal Cookies
One of the easiest cookies I know how to make - made from rolled oats, they're razor thin and lacy, golden, freckled with poppy seeds, with and anise accent from crushed fennel seeds.
Cook/prep time: PT10M/PT10M
Yield: Makes about 2 dozen cookies.
Published: 2012-12-14
http://www.101cookbo...
Ingredients
Whole Vanilla Bean Cookies
Snappy, small, fragrant, vanilla wafer cookies made with a whole vanilla pod. The entire thing!
Cook/prep time: PT15M/PT45M
Yield: Makes about 2 dozen small cookies.
Published: 2012-07-18
http://www.101cookbo...
Ingredients
Baked Quinoa Patties
A few London pics, a look at how I packed my lunch for the flight, and the baked quinoa patties I brought along with me - dill, feta, chives, cumin, and garlic.
Cook/prep time: PT25M/PT10M
Yield: Makes about a dozen patties.
Published: 2011-10-11
http://www.101cookbo...
Ingredients
Whole Wheat Chocolate Chip Skillet Cookie
Made using 100% whole wheat flour and hand-chopped chocolate chips, this is a skillet-baked twist on Kim Boyce's celebrated Chocolate Chip Cookies.
Cook/prep time: PT30M/PT10M
Yield:
Published: 2011-02-08
http://www.101cookbo...
Ingredients
Maple Huckleberry Coffee Cake Recipe
A wonderful coffee cake recipe made from fresh wild huckleberries using maple syrup as a sweetener instead of white sugar.
Cook/prep time: /
Yield: Serves 12 - 16 modest slices.
Published: 2008-09-22
http://www.101cookbo...
Ingredients
Cherry Cobbler Recipe
A rustic, cherry cobbler recipe made from fresh cherries - though you can certainly try this recipe with other types of summer fruit and berries.
Cook/prep time: /
Yield: Serves about 8.
Published: 2008-07-16
http://www.101cookbo...
Homemade and All-natural Thin Mint Recipe
As promised, an all-natural Thin Mint recipe for you. No shortening, no trans-fats from partially hydrogenated vegetable oils - just good old-
Lilikoi Passionfruit Curd Cake Recipe
A lilikoi curd-swirled loaf cake. It moist in the middle, sliceable, and delicious. Change curds with the seasons or as you are inspired - any flavor will work.
Great Chocolate Chip Cookie Recipe
If you like a serious chocolate chip cookie, this is the recipe. A high chip to dough ratio guarantees lots of chocolate in every bite, and the walnuts add crunch, density, and a
A Nice Berry Pie
My go-to berry pie recipe - a rye flour crust and mixed berry filling seasoned with fresh thyme leaves.
Cook/prep time: PT45M/PT90M
Homemade Celery Salt
Homemade celery salt couldn't be easier to make. I keep a bottomless jar of it on hand and use it on just about everything - eggs, soups, salads. It's hard to go wrong
White Whole Wheat Pizza Dough Recipe
Fantastic Peter Reinhart inspired rustic, thin-ish crust pizza dough recipe - made from white whole wheat flour, yeast, olive oil, salt, and ice water.
Peter Reinhart's Napoletana Pizza Dough Recipe
Peter Reinhart's Napoletana pizza dough recipe. It makes my all-time favorite pizza dough using a delayed-fermentation method.

File Edit View History Bookmarks Tools Help
FER Cookbook
127.0.0.1:5173/recipes/85
70%
My cookbook (ver03)
Recipes | Calculator
Recipe #85
Great Chocolate Chip Cookie Recipe
If you like a serious chocolate chip cookie, this is the recipe. A high chip to dough ratio guarantees lots of chocolate in every bite, and the walnuts add crunch, density, and a delicious flavor to the overall mix. They are that good.
Cook/prep time: /
Yield:
Published: 2005-04-11
http://www.101cookbo...
Ingredients

# Provide/inject

- https://v3.vuejs.org/guide/component-provide-inject.html
- Ako imamo duboku hijerarhiju komponenti i želimo poslati svojstvo (*prop*) ne samo neposrednom djetetu onda to može biti zamorno („štafeta”)
- *provide*: roditelj pruža svojstvo svim svojim nasljednicima (cijelo podstablo)
- *inject*: dijete prijavi što koristi
- *„long-range props”*:
  - Roditelji ne moraju znati tko koristi
  - Djeca ne moraju znati odakle dolazi
- Moguće i App-level Provide

# Emitiranje događaja

- „Suprotni smjer” – ponekad je potrebno iz komponente nešto javiti roditelju
- Dodajmo u recipe-card gumb za brisanje recepta
  - Komponenta **emitira** događaj brisanja svom roditelju
  - Roditelj se povezuje na emitirane događaje i poduzima odgovarajuće radnje – ovdje izbacuje element iz polja

# Emitiranje događaja „deleteRecipe”

ver04

components/RecipeCard.vue

```
<template>
  <small-card>
    (... isto kao prije ...)
    <button class="btn btn-danger"
            @click="deleteRecipe">
            Delete
    </button>
  </small-card>
</template>
<script>
export default {
  emits: ["deleteRecipe"],
  (...)
  methods: {
    deleteRecipe() {
      this.$emit('deleteRecipe', {id: this.id});
    }
  }
};
</script>
```

Na sljedećem slajdu

1

2

views/RecipesVie

```
<template>
  <div v-if="selectedRecipe">
    <h2>Recipe #{{ id }}</h
      <recipe-card :key="
          v-bind="selectedRecipe"
          @delete-recipe="deleteRecipe"
        ></recipe-card>
  </div></div>
  <div v-else>
    <h2>All recipes ({{allRecipes.length}})</h2><hr><div>
      <recipe-card v-for="recipe in allRecipes"
        :key="recipe.id" v-bind="recipe"
        @delete-recipe="deleteRecipe"
      ></recipe-card>
  </div></div>
</template>
<script>...
methods: {...
 deleteRecipe (args) {
   this.allRecipes = this.allRecipes.filter(x => x.id !== args.id);
   if (this.selectedRecipe && this.selectedRecipe.id === args.id) {
       this.selectedRecipe = null;
   }
 }
}...
</script>
```

3

*„Like components and props, event names provide an automatic case transformation. If you emit an event from the child component in camel case, you will be able to add a kebab-cased listener in the parent”*

# Utori (*Slots*)

ver04

https://v3.vuejs.org/guide/component-slots.html

- Komponente mogu definirati mjesto (utor, slot) u koje će se ugraditi drugi sadržaj (string, HTML, druge komponente…)
- Mi ćemo definirati „karticu” – standardni element našeg korisničkog sučelja
  - u nju možemo stavljati razne sadržaje (recept, kalkulator)

components/SmallCard.vue

```
<template>
  <div>
    <slot></slot>
  </div>
</template>
<style scoped>
    div {
        border: 1px gray solid;
        border-radius: 3px;
        width: 250px;
        box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
        padding: 10px;
        margin: 5px;
    }
</style>
```

Prijavimo u main.js kao i druge komponente.
Postoje i napredne opcije,
npr. *scoped slots, named-slots*, ZOKŽNV

# Omogućimo izmjenu recepta!

- Radi jednostavnosti: samo `name` i `description`
- Gumb za izmjenu vidljiv samo kada se odabere recept
  - Dodat ćemo property `canEdit` koji će to definirati
- Komponenta mijenja izgled s obzirom na `editMode`

# Omogućimo izmjenu recepta!

- Želimo da se izmjene unutar komponente odraze na vanjske podatke odnosno selektirani recept
- Gumb za izmjenu vidljiv samo kada se odabere recept
  - Dodat ćemo prop `canEdit` koji će to definirati
- Komponenta mijenja izgled s obzirom na `editMode`

# Povežimo prvo „vanjsku” varijablu

ver05

views/recipes.vue

```
<template>
  <div v-if="selectedRecipe">
    <div>
      <recipe-card :key="selectedRecipe.id"
          v-model="allRecipes[selectedRecipeIndex]"
          @delete-recipe="deleteRecipe"
          can-edit="true"
        ></recipe-card>
      </div>
  </div>
  <div v-else>
    <div>
      <recipe-card v-for="(recipe, index)
           in allRecipes"
        :key="recipe.id"
        v-model="allRecipes[index]"
        @delete-recipe="deleteRecipe"
      ></recipe-card>
    </div>
  </div>
</template>
...
```

Nema smisla povezati na lokalnu selectedRecipe jer se to onda neće odraziti na naše polje – zato uvodimo selectedRecipeIndex da možemo odmah direktno povezati s elementom polja allRecipes[selectedRecipeIndex]

```
...
<script>
export default {
  ...
  data() {
    return {
      selectedRecipe: null,
      selectedRecipeIndex: -1,
    };
  },
  watch: {
    $route(to, from) {
      this.selectedRecipe = this.allRecipes.find( x => x.id == this.$route.params.id);
      this.selectedRecipeIndex = this.allRecipes.findIndex( x => x.id == this.$route.params.id);
    },
  },
...
```

Ovdje bi nam bili dovoljni props iz ver04 jer se predaju samo roditelj->dijete jer canEdit nije postavljen pa se ne može ni promijeniti vezana vrijednost, ali ovako je bolje jer kasnije možemo slobodno staviti canEdit – probajte!

DM6
DM7

ver05

# … zatim recipe-card

views/recipes.vue

```
<template>

  <small-card v-if="!editMode">
        (… isto kao prije …)
    <button v-if="canEdit" @click="editMode = true">
      Edit
    </button>
  </small-card>
  <div v-if="editMode" class="editForm">
      <form @submit.prevent="submitChanges">
         <input type="text" v-model="name">
         <textarea v-
model="description"></textarea>
         <button type="submit">
          Save
         </button>
         <button @click.stop="exitSingleRecipe()">
          Cancel
          </button>
      </form>
  </div>
</template>
```

Izbačeno dosta koda koji nije u fokusu, vidjeti src, jedino u read-only dijelu sad koristimo modelValue.PROP, npr. modelValue.name, modelValue.url, itd.

Save će bubble-up u form submit, a Cancel neće zbog stop modifiera

```
<script>
export default {
  emits: ["deleteRecipe", "update:modelValue"],
  props: ["modelValue", "canEdit", "zoom" ],
  data() {
    return {
      editMode: false,
      name: this.modelValue.name,
      description: this.modelValue.description
    }
  },
  methods: {
    exitSingleRecipe() {
      this.$router.push({ path: '/recipes' });
    },
    submitChanges() {
      this.$emit('update:modelValue', {
        id: this.modelValue.id,
        image: this.modelValue.image,
        name: this.name,
        description: this.description,
        cookTime: this.modelValue.cookTime,
        prepTime: this.modelValue.prepTime,
        recipeYield: this.modelValue.recipeYield,
        datePublished: this.modelValue.datePublished,
        url: this.modelValue.url,
        ingredients: this.modelValue.ingredients
      });
      this.exitSingleRecipe();
    }
  …
```

## Slide 42

**DM6** kuzim, ali bas mi je update:modelValue  grd, moze se spomenuti state model. U biti je li bitno za recipe promjena u kartici?
Danijel Mlinarić; 15.9.2021.

**DM7** sad vidim sljedeći slajd spominjes stanja :)
Danijel Mlinarić; 15.9.2021.

# Može i bolje…

- Što se dogodi s promjenama ako odemo na Calculator, pa se vratimo na Recipes?

# Stanje aplikacije (*state*)

# Stanje

- Stanje = trenutni radni skup podataka aplikacije
- Može biti:
  A. **Lokalno stanje** (komponenti, stranica,…)
     - Odnosi se na samo jednu komponentu
     - Npr. logs u Calculator, isEdit u RecipeCard, allRecipes u Recipes
  B. **Globalno stanje**
     - Odnosi se na (potrebno u) više komponenti
     - Npr. sadržaj košarice kod kupovine, prijavljeni korisnik i uloge
     - Je li allRecipes potreban na više mjesta?
- Tipično u aplikaciji imamo i globalno i lokalno stanje

# Kako ostvariti globalno stanje? (1/2)

- *Provide/inject*?
- Moguće, ali:
  - U korijensku komponentu bi „nagurali” previše koda
  - Teže upravljivo, teži razvoj
  - Nisu jasni obrasci korištenja, izmjene
  - Podložnije pogreškama
  - Prikladno za manje aplikacije

# Kako ostvariti globalno stanje? (2/2)

- FTSE ☺: uvodimo vanjski entitet koji se bavi globalnim stanjem
- *Global singleton*
- Provodi nekakav **obrazac** upravljanja stanjem:
  - **Strogo definirani načini korištenja**, čitanja i mijenjanja stanja
  - Smanjuje mogućnost pogreške
  - Olakšava razvoj (devtools, debugging, time-travel)
- Npr.:
  - Redux, MobX
  - VueX, Pinia
  - NgRx

https://v3.vuejs.org/guide/state-management.html#simple-state-management-from-scratch

# VueX

- „**state management pattern + library** for Vue.js applications”
- „strogo definirani način korištenja” (obrazac):

https://vuex.vuejs.org/#what-is-a-state-management-pattern

# Pinia

- Zvanična podrška od strane Vue razvojnog tima

https://betterprogramming.pub/testing-pinia-is-vuex-out-43e0531824f5

# Pinia counter app – rudimentarni primjer

pinia-counter

stores/counter.js

```
import { defineStore } from "pinia";
export const useCounterStore = defineStore({
  id: "counter",
  state: () => ({
    counter: 0,
  }),
  getters: {
    count: (state) => state.counter,
    doubleCount: (state) => state.counter * 2,
  },
  actions: {
    increment() {
      this.counter++;
    },
    decrement() {
      this.counter--;
    },
  },
});
```

Count: 1

Double count: 2

+ -

state, getters i actions

App.vue

```
<template><div>
    <h1>Count: {{ count }}</h1>
    <h1>Double count: {{ doubleCount }}</h1>
    <p>
      <button @click="increment">+</button>
      <button @click="decrement">-</button>
    </p>
</div></template>
<script>
import { useCounterStore } from "./stores/counter";
import { mapState, mapActions } from "pinia";
export default {
  // data() { return { store: useCounterStore(), }; },
  computed: {
    ...mapState(useCounterStore, ["count",
"doubleCount"]),
  },
  methods: {
    ...mapActions(useCounterStore, ["increment",
"decrement"]),
    // decrement() { this.store.decrement(); },
  },
};
</script>
```

Može i komentirana verzija, ali elegantnije je koristiti helpere mapState i mapActions koji će nam zadana svojstva/funkcije mapirati unutra komponente

# Dodajmo prijavljenog korisnika!

ver06

- Plan:
  - Store sadrži ime prijavljenog korisnika
  - Napraviti ćemo novu stranicu `LoginView.vue` za prijavu
  - U router/index.js:
    - Ako **nije** prijavljen -> `/login`
    - Ako je prijavljen – kao prije
  - Store ćemo registrirati i on će biti vidljiv svim komponentama, u našem slučaju trebaju ga `router`, `LoginView` i `App` (radi ikonice gore desno)
  - **Store pravila:**
    - Čitamo pomoću **getters**
    - Mijenjamo stanje pomoću **actions**

# Store

ver06

stores/index.js

```
import { defineStore } from "pinia";

export const useAuthStore = defineStore("auth", {
  state: () => ({
    // ovo bi zapravo bio neki objekt, ali ovdje
radi jednostavnosti je user=username
    user: null,
    _landingUrl: "/",
  }),
  getters: {
    isAuthenticated: (state) => !!state.user,
    username: (state) => state.user,
    landingUrl: (state) => state._landingUrl,
  },
  actions: {
    setUsername(username) {
      this.user = username;
    },
    setLandingUrl(url) {
      this._landingUrl = url;
    },
  },
});
```

main.js

```
...
import { createApp } from "vue";
import { createPinia } from "pinia";
...
const app = createApp(App);

app.use(createPinia());
app.use(router);
```

Stvaramo singleton i vežemo ga uz aplikaciju. Kasnije, u komponentama, ćemo ga jednostavno referencirati s:

```
import { useAuthStore } from "../stores/auth";
...
const auth = useAuthStore();
```

# Router

ver06

Router/index.js

```
import { useAuthStore } from "../stores/auth";
const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: "/login",
      name: "login",
      component: LoginView,
    },
    {
      path: "/",  name: "recipes", component: RecipesView,
    },
...
router.beforeEach(async (to) => {
  // redirect to login page if not logged in and trying to access
a restricted page
  const publicPages = ["/login"];
  const authRequired = !publicPages.includes(to.path);
  const auth = useAuthStore();
  if (authRequired && !auth.isAuthenticated) {
    auth.setLandingUrl(to.fullPath);
    return "/login";
  }
});
```

Dodajemo novu rutu na stranicu za prijavu korisnika (sljedeći slajd)

„Lovimo” sve adrese i prije usmjeravanja provjeravamo je li potrebna prijava

Pamtimo traženi URL da bi mogli korisnika preusmjeriti nakon uspješne prijave

# UserLogin.vue

tag: v7.0

views/LoginView.vue

```vue
<template>
    <small-card>
      <h3>Dear unknown user, please login</h3>
      <form @submit.prevent="login">
        <div class="form-group row">
          <label class="col-4 col-form-label">Login</label>
          <div class="col-8">
            <input class="form-control"
              placeholder="username"
              v-model.trim="username"
            />
          </div>
        </div>
        <div class="d-flex justify-content-center">
          <button class="btn btn-success m-3"
            type="submit">
            Let me in!
          </button>
        </div>
      </form>
    </small-card>
</template>
```

```vue
<script>
import { mapActions, mapState } from "pinia";
import { useAuthStore } from "../stores/auth";
export default {
  data() {
    return {
      username: "",
    };
  },
  computed: {
    ...mapState(useAuthStore, ["landingUrl"]),
  },
  methods: {
    ...mapActions(useAuthStore, ["setUsername"]),
    login() {
      // sve puštamo, naravno inače treba provjeriti
      this.setUsername(this.username);
      this.$router.push({ path: this.landingUrl });
    },
  },
};
</script>
```

Vraćamo korisnika na inicijalno zatraženi URL

# Izmjene ostatka aplikacija – drugi store *recipes*

- Pinia jednako tretira/dopušta sinkrone i asinkrone funkcije pa ćemo tako u recipe store premjestiti
  - Dohvat recepata (**asinkrona** funkcija)
  - Brisanje i ažuriranje recepata (sinkrone funkcije)
- Promijenit ćemo ustroj i povezivanje `Recipes` i `RecipeCard`:
  - **prije** su bili spregnuti putem v-model, emit, refresh iz `Recipes`
  - **sada** obje imaju pristup storeu i `Recipe` predaje samo `id`

# Pogledajmo prvo *Recipe store*:

ver06

store/recipes.js

```
export const useRecipeStore = defineStore("recipes", {
  state: () => ({
    _allRecipes: [],
  }),
  getters: {
    allRecipes: (state) => state._allRecipes || [],
    getRecipeById: (state) => {
      return (id) => state._allRecipes.find((rcp) => rcp.id === id);
    },
  },
  actions: {
    async refreshRecipes() {
      if (this.allRecipes.length === 0) {
        try {
            ...
            this._allRecipes = recipes.slice(0, 50);
          } else { ... }
        } catch (error) {...}
      }
    },
    deleteRecipe(id) {
      this._allRecipes = this._allRecipes.filter((x) => x.id !== id);
    },
    updateRecipe(recipe) {
      this._allRecipes = this._allRecipes.map((x) =>
        x.id == recipe.id ? recipe : x
      );
    },
  },
});
```

Trebat će nam za RecipesView

Trebat će nam za RecipeCard, primjer gettera s argumentima

*Getters are just computed properties behind the scenes, so it 's not possible to pass any parameters to them. However, you can return a function from the getter to accept any arguments:*

Trivijalno keširanje, razmisliti o opcijama sinkronizacije s *backendom*... Svakako dodati boolean da se može forsirati refresh...

# … zatim RecipeCard - on radi samo na temelju id i canEdit (i veze sa Storeom), mapActions, mapState:

ver06

components/RecipeCard.js

```
export default {
  emits: ["recipeDeleted", "recipeUpdated"],
  props: ["id", "canEdit", "zoom"],
  data() { return {
      editMode: false, recipe: { url: "" },
    }; },
  computed: {
    ...mapState(useRecipeStore, ["getRecipeById"]),
    idUrl() { return "/recipes" + this.id; },
  },
  methods: {
    ...mapActions(useRecipeStore, ["deleteRecipe", "updateRecipe"]),
    deleteRecipeLocal() {
      this.deleteRecipe(this.id); this.$emit("recipeDeleted", { id: this.id });
      this.exitSingleRecipe();
    },
    exitSingleRecipe() { this.$router.push({ path: "/recipes" }); },
    submitChanges() {
      this.updateRecipe(this.recipe); this.$emit("recipeUpdated", this.recipe);
      this.exitSingleRecipe();
    },
  },
  async created() {
    this.recipe = { ...this.getRecipeById(this.id) };
  },
};
```

Template je isti, samo sada koristimo recipe.property, npr.

```
<span class="badge badge-secondary">Yield: {{ recipe.recipeYield }}</span>
```

Također, u edit formi smo izbacili vlastite varijable već vežemo direktno na recipe.name i recipe.description – to je u redu jer je taj recipe klonirani element polja iz allRecipes, dakle lokalna kopija, npr.

```
… <div v-if="editMode" class="editForm">
      <form @submit.prevent="submitChanges">
      …
          <input type="text" class="form-control" id="name" placeholder="name"  v-model="recipe.name">
```

Ljubaznost: možda će post-festum zanimati onoga tko me pozvao. Primijetiti promjenu naziva eventa - sad je pasiv

Inicijalno prazan, korisnik to neće ni vidjeti, ali na jedno mjestu koristimo url.substring, pa da ne bi bilo undefined.substring

Kloniramo putem spread operatora.
Mogli smo i u mounted, ali created je ranije u lifecycleu

# … i konačno - RecipesView

ver06

views/RecipesView

```
export default {
  props: ["id"],
  data() {
    return { selectedRecipe: null };
  },
  computed: {
    ...mapState(useRecipeStore, ["allRecipes"]),
  },
  methods: {
    ...mapActions(useRecipeStore, ["refreshRecipes"]),
    recipeUpdated(recipe) {
      console.log("So now i know recipe is updated...", recipe)
    },
    recipeDeleted(args) {
      if (this.selectedRecipe
          && this.selectedRecipe.id === args.id) {
        this.selectedRecipe = null;
      }
    },
  },
  async mounted() {
    await this.refreshRecipes();
    this.setSelectedRecipe();
  },
};
```

Recipes se bitno pojednostavnio – sad se brine samo time je li recept odabran ili ne i kako to prikazati

Ažuriranje nas zasad ni ne zanima, ali brisanje da – koristimo „ljubaznost" odnosno dojavu iz RecipeCard

```
<div v-if="selectedRecipe">
  <h2>Recipe #{{ id }}</h2><br>
  <div class="d-flex justify-content-center">
    <recipe-card :key="selectedRecipe.id"
        :id="selectedRecipe.id"
        @recipe-updated="recipeUpdated"
        @recipe-deleted="recipeDeleted"
        can-edit="true"
      ></recipe-card>
    </div>
</div>
<div v-else>
  <h2>All recipes ({{$store.getters.allRecipes.length
}})</h2>
  <hr>
  <div class="container-fluid p-2 d-flex flex-wrap">
    <recipe-card v-for="recipe in allRecipes"
      :key="recipe.id"
      :id="recipe.id"
      @recipe-updated="recipeUpdated"
      @recipe-deleted="recipeDeleted"
    ></recipe-card>
  </div>
</div>
```

DZ: promijeniti template da pokazuje spinner dok recepti još nisu dohvaćeni…

# I još jedna stvar…

## npm run build

```
\  Building for production...

 WARNING  Compiled with 1 warning                                    4:46:18 PM
 warning

asset size limit: The following asset(s) exceed the recommended size limit (244 KiB).
This can impact web performance.
Assets:
  img/disco.b921cfd5.gif (2.06 MiB)

  File                                  Size              Gzipped

  dist\js\chunk-vendors.a68e7e1d.js     142.14 KiB      50.69 KiB
  dist\js\app.a448ee5c.js               15.79 KiB        5.00 KiB
  dist\js\chunk-2d0ddf93.8177cced.js    0.48 KiB         0.33 KiB
  dist\js\chunk-91eb2748.2e907fc5.js    0.42 KiB         0.30 KiB
  dist\css\app.1961192e.css             0.52 KiB         0.34 KiB

  Images and other types of assets omitted.

 DONE  Build complete. The dist directory is ready to be deployed.
 INFO  Check out deployment instructions at https://cli.vuejs.org/guide/deployment.html
```

# Razvojna i produkcijska okolina

- U razvojnoj okolini:
  - Pokreće se lokalni web-poslužitelj
  - Prevodi se .vue kontinuirano kod svakog snimanja
  - Hot-Module-Replacement (HMR)
  - Datoteke nisu minimizirane
  - Radni okvir daje raskošne provjere i upozorenja
  - Itd.
- U produkcijskoj okolini:
  - SPA je zapravo „samo” HTML + CSS + JS koji zatim trebamo poslužiti pregledniku (static web hosting)
  - Minifikacija HTML+CSS+JS datoteka (+ source maps)
  - Vendor chunk splitting (razdvojiti naš kod od knjižnica)
  - Optimizacija koda
  - Optimizacija učitavanja?

# Optimizirajmo učitavanje (dijelova) stranice

- Jedan od nedostataka SPA je veliki inicijalni zahtjev
- Rješenje: učitajmo dijelove aplikacije kad (i ako trebaju)!
- Dodajmo u našu aplikaciju besmislenu „Disco” komponentu i stranicu:

- Učitajmo ju asinkrono – kad (ako) nam zatreba!

# Komponenta i stranica su trivijalne:

ver07

views/DiscoView.vue

```
<template>
  <div>
    <h2>Disco!!!</h2>
    <div class="container-fluid p-2 d-flex justify-content-center">
      <disco-component></disco-component>
    </div>
  </div>
</template>
```

components/DiscoComponent.vue

```
<template>
  <div>
    <img src="../assets/disco.gif"/>
  </div>
</template>
```

# Prvo učitajmo komponentu asinkrono…

ver07

main.js

```
import { createApp, defineAsyncComponent } from "vue";
import App from "./App.vue";
import store from "./store";
import router from "./router";
import RecipeCard from './components/RecipeCard.vue';
import TempConverter from './components/TempConverter.vue';
import SmallCard from './components/SmallCard.vue';

const DiscoComponent = defineAsyncComponent(() => import('./components/DiscoComponent.vue'));

const app = createApp(App);
app.use(store);
app.use(router);
app.component('recipe-card', RecipeCard);
app.component('temp-converter', TempConverter);
app.component('small-card', SmallCard);
app.component('disco-component', DiscoComponent);
app.mount("#app");
```

# ...potom i stranicu

ver07

router/index.js

```
import { defineAsyncComponent} from "vue";
import {  createRouter,  createWebHistory} from "vue-router";
import Recipes from "../views/Recipes.vue";
import Calculator from "../views/Calculator.vue";
import NotFound from "../views/NotFound.vue";
// u routeru ne koristiti defineAsyncComponent
const Disco = () => import('../views/DiscoView.vue');
const routes = [
  {
    path: "/",
    component: Recipes,  // mogli smo i redirect: "/recipes"
  },...
  {
    path: "/disco",
    component: Disco,
  },
  ...
  ];
```

Ako pokrenemo i u razvojnoj okolini vidjet ćemo da se datoteke dohvaćaju kada su potrebne.
Kada napravimo production build, onda će to biti posebne chunk datoteke.

# Konačno, napravimo build

- npm run build

U razvojnoj okolini:

| Name | Method | Status | Type | Initiator | Size | T |
|---|---|---|---|---|---|---|
| app.js | GET | 200 | script | (index) | 274 kB | 1 |
| chunk-vendors.js | GET | 200 | script | (index) | 3.0 MB | 6 |

```
λ npm run build

> fer-demo-spa@0.0.0 build
> vite build

vite v3.2.5 building for production...
✓ 45 modules transformed.
dist/assets/disco.0d123211.gif            2112.26
dist/assets/travolta404.0518648b.gif      2678.9 KiB
dist/index.html                           0.70 KiB
dist/assets/DiscoComponent.4c4e8f93.js    0.22 KiB / gzip: 0.19 KiB
dist/assets/DiscoView.c1014e9c.js         0.30 KiB / gzip: 0.23 KiB
dist/assets/index.40af8dc2.css            1.99 KiB / gzip: 0.79 KiB
dist/assets/index.95112b1f.js             89.54 KiB / gzip: 35.13 KiB
```

Ovisnosti (tuđi kod) u posebnoj datoteci, naš kod u main.js, preostali dijelovi (asinkroni) našeg koda u chunk datoteka i minificirani CSS.

# Poslužimo putem naše backend aplikacije

- Kopiramo /dist direktorij koji je napravio vue-cli u /public folder backend

# Projekt

- Napraviti SPA aplikaciju u Vue3 radnom okviru
- *Peer assessment* putem Edgara
- Više u Edgaru…

# Komentar: Composition API

The setup attribute is a hint that makes Vue perform compile-time transforms that allow us to use Composition API with less boilerplate. For example, imports and top-level variables / functions declared in <script setup> are directly usable in the template.

- U primjerima je korišten jednostavniji *„Options API"*
- Počev od v2.7. moguće je koristiti alternativni *„Composition API"*
- Primjer s https://vuejs.org/guide/extras/composition-api-faq.html#what-is-composition-api

```
<script setup>
import { ref, onMounted } from 'vue'

// reactive state
const count = ref(0)

// functions that mutate state and trigger updates
function increment() {
  count.value++
}

// lifecycle hooks
onMounted(() => {
  console.log(`The initial count is ${count.value}.`)
})
</script>

<template>
  <button @click="increment">Count is: {{ count }}</button>
</template>
```

# Zašto *Composition API*?

- Better Logic Reuse
  - Primarna prednost: ponovna uporaba programske logike – odatle composition
  - Pogledati npr. https://vueuse.org/
- More Flexible Code Organization
  - Slika desno ->
- Bolji type inference, ts.
- Efikasniji i manji konačni kod
- **Nota bene:**
  - **Mogu se kombinirati Options + Comoposition**
  - **Options neće biti deprecated!**

Options API

Composition API

# Composition API

- Nećemo ga detaljno objašnjavati dalje
- U projektu možete koristiti bilo koji, a mogu se i kombinirati
- Za one koji se žele znati više o Composition API-ju, pogledati:
  - Prerađenu verziju konačnog primjera ver07-composition-api
  - Vue dokumentaciju, podesite *API preference*:

# Napredni razvoj programske potpore za web

**- predavanja -**

**2022./2023.**

## 10. Atributi kvalitete web-aplikacija (i kako ih poboljšati…)

# Creative Commons

- slobodno smijete:
  - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
  - **prerađivati** djelo
- pod sljedećim uvjetima:
  - **imenovanje:** morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
  - **nekomercijalno:** ovo djelo ne smijete koristiti u komercijalne svrhe.
  - **dijeli pod istim uvjetima:** ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

*U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela.*
*Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.*
*Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.* *Tekst licence preuzet je s http://creativecommons.org/*

of visits are abandoned if a mobile site takes longer than 3 seconds to load.

Google Data, Global, n=3,700 aggregated, anonymized Google Analytics data from a sample of mWeb sites opted into sharing benchmark data, March 2016.

https://www.thinkwithgoogle.com/consumer-insights/consumer-trends/mobile-site-load-time-statistics/

# Zahtjevi na sustav

1. **Svojstva sustava, funkcionalni** zahtjevi (*system features – functional reqs*)
   - Definiraju funkcionalnost (nekog dijela) sustava, apstrahirano output = f(input)
   - Npr. sustava mora omogućiti pretraživanje hrane, odabir hrane (dodavanje u košaricu), predavanja i plaćanje narudžbe…
   - Koristimo: use-cases, user-flows…
   - Općenito: bilo koja arhitektura može ostvariti bilo koja funkc. svojstva ☹
2. **Atributi kvalitete, nefunkcionalni** zahtjevi (*quality attributes, non-functional reqs*)
   - Svojstva koja sustava treba imati, npr. dostupnost, skalabilnost, pouzdanost, sigurnost, …
   - Jako utječu na arhitekturu sustava
3. **Zadana ograničenja** sustava (*system constraints*)
   - Tehnička, poslovna, pravna
   - Npr. ograničen budžet, ograničen broj i znanje djelatnika, licence, vremenski rokovi, itd.

U nastavku se bavimo samo s:

# 2. nefunkcionalni zahtjevi

*"Quality is never an accident;*
*it is always the result of high intention, sincere effort, intelligent direction and skillful execution;*
*it represents the wise choice of many alternatives."*

***William A. Foster***

# Atributi kvalitete - motivacija

- Koji su razlozi za redizajn/prepravak sustava?
  - Sustav je **spor**
  - Nije **skalabilan**
  - Težak za **održavanje**
  - Ima **sigurnosne** propuste
  - Spor **razvoj**
  - **Preskup**
  - …
- Redizajnom sustava ne mijenjamo funkcionalnost
  - Neke od atributa korisnici osjećaju (npr. brz/spor), a neke ne (teškoće razvoja i održavanja)

# Atributi kvalitete

- Atribut kvalitete je **mjerljivo** ili **provjerljivo** svojstvo sustava koje govori koliko dobro sustav **zadovoljava** potrebe s obzirom na neku **dimenziju kvalitete.**
- Izravno utječe na arhitekturu sustava (i obratno)
- Na primjer:
  - Sustav mora vratiti odgovor u tražilici za manje od 2s
  - Sustav mora biti raspoloživ barem 99.9% vremena
- „Mjerljivo ili provjerljivo”:
  - Sustav mora vratiti odgovor unutar 2s
  - ~~Sustav mora brzo vratiti odgovor~~
- Neki atributi kvalitete su u suprotnosti s drugima (npr. sigurnost je tipično u suprotnosti sa „svima”), tako da ne možemo maksimizirati po svima
  - Tradeoff ☹

# Atributi kvalitete

1. *Performance* (učinkovitost)
2. *Scalability* (skalabilnost)
3. *Availability* (dostupnost)
4. …

## Quality attributes [ edit ]

Notable quality attributes include:

- accessibility
- accountability
- accuracy
- adaptability
- administrability
- affordability
- agility (see Common subsets below)
- auditability
- autonomy [Erl]
- availability
- compatibility
- composability [Erl]
- confidentiality
- configurability
- correctness
- credibility
- customizability
- debuggability
- degradability
- determinability
- demonstrability
- dependability (see Common subsets below)
- deployability
- discoverability [Erl]
- distributability
- durability
- effectiveness
- efficiency
- evolvability
- extensibility
- failure transparency
- fault-tolerance
- fidelity
- flexibility
- inspectability
- installability
- integrity
- interchangeability
- interoperability [Erl]
- learnability
- localizability
- maintainability
- manageability
- mobility
- modifiability
- modularity
- observability
- operability
- orthogonality
- portability
- precision
- predictability
- process capabilities
- producibility
- provability
- recoverability
- redundancy
- relevance
- reliability
- repeatability
- reproducibility
- resilience
- responsiveness
- reusability [Erl]
- robustness
- safety
- scalability
- seamlessness
- self-sustainability
- serviceability (a.k.a. supportability)
- securability (see Common subsets below)
- simplicity
- stability
- standards compliance
- survivability
- sustainability
- tailorability
- testability
- timeliness
- traceability
- transparency
- ubiquity
- understandability
- upgradability
- usability
- vulnerability

https://en.wikipedia.org/wiki/List_of_system_quality_attributes

# 1. Preformance / response time

- *"It's about time!"*
- *Response time (end-to-end latency)* - vrijeme odziva
- **Vrijeme potrebno da dobijemo odgovor**

**RT = *waiting time (latency) + processing time***

  - **WT** - "putovanje" kroz komunikacijske kanale, čekanje u redovima, itd.
  - **PT** - vrijeme provedeno u "našem kodu" na serveru
- RT se nekad naziva i *End-to-end latency*

# Paziti kod mjerenja - izmjeriti oboje!

- Klasična pogreška: mjeriti samo processing time na serveru, tada:
  - (neka je PT1 = PT2 = 2s)
  - AVG("RT") = 2.0s

- Treba mjeriti iz perspektive klijenta, tada:
  - RT1 = 2s + 2s = 4s
  - RT2 = 3.5s + 2s = 5.5s
  - AVG(RT) = 4.75s

# RT -> koje mjere koristiti?

- Primjer, RT s tri poslužitelja:

- Medijan 2262.6ms
- Prosjek 2702.9ms

dobra metrika?

# RT -> histogram (distribucija RT-a)

- N-ti **percentil** nekog skupa podataka je vrijednost koja je veća od n% vrijednost iz skupa
  - 50. percentil = median
  - 25. percentil = prvi kvartal, itd.
- Pretvorimo sad histogram u **percentile** –> sljedeći slajd

# Percentili *i Tail latency* (1)

- TL je mali postotak RT-a koji traje najdulje (u usporedbi s većinom):
  - Svakako > max(prosjek, medijan)
  - Npr. „zadnjih 10%”

# Percentili *i Tail latency* (2)

- Idealno - što kraći „rep”
- Ciljeve si onda možemo bolje postaviti koristeći percentile (a ne prosjek i medijan), npr.:
  - RT < 1000ms na 95. percentilu
  - RT < 1500 na 99. percentilu
  - Itd.

# *1. Preformance / Throughput* - propusnost

- Podjednako važna metrika
- Dva pogleda/definicije:

1. Broj poslova koji se mogu obaviti po jedinici vremena (mjereno u broj/s)
2. Količina podataka koja se može procesirati/ jedinici vremena (mjereno u {k|M|G|…}B/s)

# Degradacija performansi?

- „Nešto” je usko grlo:
  - CPU?
  - Memorija?
  - Previše konekcija?
  - …

# 2. Skalabilnost (1)

- Promet (~opterećenje) nikad nije jednolik
  - Dnevne fluktuacije
  - Sezonske fluktuacije
  - …
  - Organski rast
- Analogija: autoput

Neke definicije:

- Skalabilnost je svojstvo sustava da se nosi s rastućom količinom posla dodavanjem resursa sustavu.

  Bondi, André B.

  - Skalabilnost opisuje sposobnost rukovanja većim brojem korisnika, klijenata, podataka, transakcija ili zahtjeva bez utjecaja na korisničko iskustvo
  - Učinkovito (*efficient*) ili ekonomično (*cost-efficient, cost-effective*) ispunjavanje zahtjeva

# Još neke definicije skalabilnosti… (2)

- *Capacity of the system to handle increasing amount work without affecting existing system.*
- *Scalability is a characteristic of a system, model or function that describes its capability to cope and perform under an increased or expanding workload. A system that scales well will be able to maintain or even increase its level of performance or efficiency when tested by larger operational demands.*
- *The measure of a system's* ***ability to increase or decrease*** *in performance and cost in response to changes in application and system processing demands.*
- *Property of software and hardware systems that can improve functionality, performance, and speed by adding or removing more processors, memory, and other resources.*

*https://www.igi-global.com/dictionary*

# 2. Skalabilnost (3)

- Idealno (~nemoguće) linearno:

# 2. Skalabilnost (4)

- **Tri vrste** skalabilnosti:

1. **Vertikalna**
   - Dodavanje resursa **jednom računalu**
2. **Horizontalna**
   - Dodavanje novih instanci (resursa) u klaster serverskih računala
3. **Organizacijska**
   - Povećanje produktivnost dodavanjem više radnika

https://www.webairy.com/horizontal-and-vertical-scaling

**Značajke važne za ispitivanje skalabilnosti**

# 2. Skalabilnost: vertikalna vs horizontalna (5)

## Vertikalna

- ✓ Ekonomično – dodavanje memorije ili procesora, diska na postojeće računalo je jeftinije nego kupovina novog
- ✓ Jednostavnija i brža komunikacija među procesima
- ✓ Jednostavnije održavanje
- ✓ **Ne treba mijenjati kod! Vrijedi za „sve" aplikacije**
- ✓ **Migracija i scale up/down je lagan (posebice u oblaku)**
- X **SPoF**
- X **Veća vjerojatnost prekida u radu**
- X **Ograničenih mogućnosti ☹**

## Horizontalna

- ✓ Logički jednostavnije (ne treba detaljno tražiti usko grlo)
- ✓ Manja vjerojatnost prekida rada
- ✓ **Povećana izdržljivost i otpornost na pogreške**
- ✓ **Bolje performanse!**
- ✓ **„neograničenog" rasta**
- X **Promjena koda**
- X **Povećana složenost sustava**
- X Veći inicijalni troškovi

# 2. Skalabilnost: organizacijska skalabilnost (6)

- Što ako na i sam **razvojni proces** gledamo kroz prizmu skalabilnosti?
  - Rastući posao = razvijanje/programiranje sustava
  - Dodatni resursi = razvojni inženjeri
- Hoćemo li imati linearnu skalabilnost?
  - Veći koordinacijski/organizacijski troškovi
  - Različita znanja
  - *Code merge* konflikti
  - …
  - „sto babica – kilavo dijete”
- Visoko centralizirane arhitekture i centralizirani programski kôd će slabije skalirati
  - Razdvojiti na module/servise/…
  - Manji timovi koji se bave svojim poslom – *separation of concerns*

# 3. Availability - motivacija

- *This past Tuesday morning Pacific Time an Amazon Web Services engineer was debugging an issue with the billing system for the company's popular cloud storage service S3 and accidentally mistyped a command. What followed was a several hours' long cloud outage that wreaked havoc across the internet and resulted in hundreds of millions of dollars in losses for AWS customers and others who rely on third-party services hosted by AWS.*
- https://www.datacenterknowledge.com/archives/2017/03/02/aws-outage-that-broke-the-internet-caused-by-mistyped-command
- https://awsmaniac.com/aws-outages
- https://github.com/danluu/post-mortems

# 3. Dostupnost (*Availability*) (1)

- Spremnost sustava da obavlja svoje zadaće
- Jako veliki utjecaj na korisnike!
  - Gubitak novaca i klijenata
- **Dostupnost je vjerojatnost da će sustav biti dostupan (raditi) u nekom proizvoljnom trenutku** (kada se koristi pod navedenim uvjetima u idealnom okruženju)
- ***Uptime*** = vrijeme kad je sustav (bio) dostupan
- ***Downtime*** = vrijeme kad je sustav (bio) nedostupan

$$A = U / (U + D)$$

- Tipično se izražava postotno
  - Ako se *uptime* izrazi postotno onda A = U

# 3. Dostupnost (*Availability*) (2)

- Alternativna i slična definicija koja je zgodna kad imamo puno različitog hardvera i heterogene sustave (cloud, …)

https://en.wikipedia.org/wiki/Availability

- MTBF - *Mean Time Between Failures* - predviđeno prosječno vrijeme između dva kvara
  - Za popravljive sustave
- MTTF - *Mean Time To Failure* - predviđeno prosječno vrijeme do kvara
  - Za nepopravljive sustave
- MTTR - *Mean Time To Recovery* - predviđeno prosječno vrijeme potrebno da se otkrije i popravi pogreška

$$A = MTBF / (MTBF + MTTF)$$

→0 ?

# 3. Dostupnost (*Availability*) (3)

- Koja nam je ciljana dostupnost? 99.9% +
- https://sre.google/sre-book/availability-table/#tablea1

| | per year | per quarter | per month | per week | per day | per hour |
|---|---|---|---|---|---|---|
| 90% | 36.5 days | 9 days | 3 days | 16.8 hours | 2.4 hours | 6 minutes |
| 95% | 18.25 days | 4.5 days | 1.5 days | 8.4 hours | 1.2 hours | 3 minutes |
| 99% | 3.65 days | 21.6 hours | 7.2 hours | 1.68 hours | 14.4 minutes | 36 seconds |
| 99.5% | 1.83 days | 10.8 hours | 3.6 hours | 50.4 minutes | 7.20 minutes | 18 seconds |
| 99.9% | 8.76 hours | 2.16 hours | 43.2 minutes | 10.1 minutes | 1.44 minutes | 3.6 seconds |
| 99.95% | 4.38 hours | 1.08 hours | 21.6 minutes | 5.04 minutes | 43.2 seconds | 1.8 seconds |
| 99.99% | 52.6 minutes | 12.96 minutes | 4.32 minutes | 60.5 seconds | 8.64 seconds | 0.36 seconds |
| 99.999% | 5.26 minutes | 1.30 minutes | 25.9 seconds | 6.05 seconds | 0.87 seconds | 0.04 seconds |

*"Three nines"*

*"Four nines"*

*"Five nines"*

Detaljnija tablica: https://en.wikipedia.org/wiki/High_availability

# Kako postići visoku dostupnost (*High Availability*)?

- -> Otpornošću na pogreške (***Fault tolerance***)
- Otpornost na pogreške je svojstvo koje omogućuje sustavu da nastavi ispravno raditi i biti dostupno u slučaju kvara jedne ili više njegovih komponenti.

https://en.wikipedia.org/wiki/Fault_tolerance

- Koje pogreške?
  - Ljudska pogreška (vidi post mortems)
  - Softverske greške
  - Hardverske greške

# Kako postići HA/FT?

- **Tri načela**:

1. Elimininiranje SPoF :
   - Replikacija
   - Redundancija
   - Potrebno je imati "*reliable crossover*", pouzdan rezervni sustav jer on može postati SPOF
2. Uočavanje pogrešaka (i izolacija pokvarenih komponenti)
   - Sustav za nadgledanje (npr. Prometheus)
     - False positives (ok) i false negatives (!ok)
3. Oporavak od pogreške:
   - Izbacivanje komponente
   - Ponovo pokretanje komponente
   - Rollback na zadnje ispravno stanje

# SLA, SLO, SLI

- ***Service Level Agreement*** je ugovor (pravni dokument) između jednog ili više pružatelja usluga i jednog primatelja usluge koji definira razne aspekte usluge - kvalitetu usluge, dostupnost, dužnosti, itd.
  - Sadrži N SLO-a
  - Definira i (financijske) kazne
- ***Service Level Objective*** definira ciljanu razinu nekog atributa kvalitete, npr.:
  - Usluga će biti dostupna 99.9% vremena u razdoblju od jedne godine
  - Vrijeme odziva će u 90% slučajeva biti manje od 1.5s
  - …
- ***Service Level Indicator*** - mjera usklađenosti s SLO-ima
  - Mjerimo post-festum

https://en.wikipedia.org/wiki/Service-level_agreement

# Ali kako?

***Architecture building blocks, patterns & techniques***

U nastavku ćemo pogledati neke tehnike, obrasce i gradivne blokove koji nam omogućuju da ostvarimo ciljane razine atributa kvalitete, ponajprije **skalabilnosti**

# *Load balancing* (uravnoteženje opterećenja) (1)

- **Uravnoteženje opterećenja**: proces u kojem **raspodjeljujemo zadaće** na **skup radnika** kako bi cjelokupni proces bio **učinkovitiji**.

- *Single server abstraction*
- Prednosti:
  - *High **scalability**, auto-scaling (u oblaku)*
  - H**A** - ugasi server koji ne radi
  - Učinkovitost
    - Iako, LB dodaje malo latencije
  - ***Maintenability***, možemo ugasiti server, raditi na njemu

# *Load balancing* (uravnoteženje opterećenja) (2)

- Tipovi:

1. **DNS LB-ovi**
   - Prednost: jednostavan, robustan, brz
   - Nedostatci: ne prati radi li server, podatci mogu biti keširani/zastarjeli, strategija je uvijek jednostavna - *round robin*
2. **Hardverski i softverski LB-ovi**
   - Mogu uzeti u obzir stanje server, snagu, trenutno opterećenje, itd.
   - Može biti i mreža (hijerarhija) internih LB-ova
3. **GSLB: *Global Server Load Balancer*** = 1 + 2

   1. Ipak, pati od keširanja

https://www.edgenexus.io/blog/mr-worldwide-presents-brief-overview-global-server-load-balancing-gslb/

# LB softver - informativno

- **HAProxy** *is a free, very fast and reliable reverse-proxy offering high availability, load balancing, and proxying for TCP and HTTP-based applications. It is particularly suited for very high traffic web sites and powers a significant portion of the world's most visited ones. Over the years it has become the de-facto standard opensource load balancer, is now shipped with most mainstream Linux distributions, and is often deployed by default in cloud platforms*.

- **NGINX** - *Load balance HTTP traffic across web or application server groups, with several algorithms and advanced features like slow-start and session persistence*
  - https://www.nginx.com/blog/nginx-load-balance-deployment-models/
  - https://www.nginx.com/resources/glossary/reverse-proxy-vs-load-balancer/

- Cloud based LBs & GSLB rješenja:

  - Amazon: Amazon ELB, Amazon Route 53
  - Microsoft Azure Load Balancer & Azure Traffic Manager
  - GCP Cloud Load Balancer & Cloud DNS
  - …

# *Content Delivery Networks*

- CDN je **geografski distribuirana** mreža *proxy* servera i njihovih podatkovnih centara. Cilj je pružiti **visoku dostupnost** i **performanse** prostornom distribucijom usluge u odnosu na krajnje korisnike.

https://en.wikipedia.org/wiki/Content_delivery_network

- Strateški raspoređeni, fizički bliži korisnicima
- Keširaju: slike, video, kôd (JS, CSS, …)
  - Još dosta HTTP1.1: https://w3techs.com/technologies/history_overview/site_element/all
- Kao klijenti CDN-a, imamo dvije strategije publiciranja:
  1. ***Pull*** – CDN povlači od nas resurse i kešira ih, osvježava kad isteknu
  2. ***Push*** – sami publiciramo (*push*) na CDN promjene
     - Može se izvesti kao dugi TTL i ručno brisanje keša na CDN-u

# Higijena: kako (besplatno) smanjiti bandwidth?

- https://almanac.httparchive.org/en/2022/page-weight

**594%**

*Figure 21.7. Growth in mobile page weight over 10 years*

Show description of Figure 21.8

*Figure 21.8. Median page weight by content type.*

# Higijena: kako (besplatno) smanjiti bandwidth?

- **Kôd**: smanjiti (*minify*) JS i CSS, smanjiti broj datoteka (HTTP zahtjeva)
- **Slike**: odabrati pravi format, više rezolucija (<picture>), *lossy/lossless*, alati za optimizaciju slika
  - Kad png, kad jpg, koja je razlika?
- **Za sve** što može: offload na CDN

- HTTP:
  - Content-Encoding: gzip
    - Lagano – „uključiti”
    - Ne koristiti za slike i video, tj. već komprimirane formate
  - Cache
    - Što keširati? Ne: HTML, Da: slike, video, JS+CSS, …
    - → Sljedeći slajd

# HTTP cache

- *Cache* se općenito može nalaziti **na tri mjesta**:
  1. Na pregledniku (private)
  2. (ako postoji) Na N proxy/CDN poslužitelja (public)
  3. (ako postoji) Na „našem” *reverse proxy* poslužitelju
     - Često ujedno i LB
     - Npr. Varnish ili NGINX

# HTTP Cache Headers

## Klijent- Request (Validacija)

- `If-None-Match` ⟷ `ETag`
- `If-Modified-Since` ⟷ `Last-Modified`

## Server - Response

- `expires`
- `pragma`
- `Cache-Control: <directive> [, <directive>]*`
- **Directives**:
- **Public**: This directive indicates that the response can be stored by any cache without any restriction. In case the response is non-cacheable, it can still be cached.
- **Private**: It indicates that only browser cache is eligible to store the response.
- **no-cache**: It indicates that the response can be stored by any cache without any restriction even if it is non-cacheable. The condition that needs to be satisfied here is that the stored response must be validated by the origin server before being used.
- **no-store**: It indicates that the response cannot be stored by any cache.
- **max-age=<seconds>**: It indicates the maximum amount of time a resource remains fresh and can be requested to be accessed.
- …

https://www.geeksforgeeks.org/http-headers-cache-control/

# *Building block: Message Broker*

- **Asinkrona** komunikacija, kada je zahtjev **zahtjevan** ☺
- *Message Broker* (posrednik poruka):
  - Pohranjuje poruke između pošiljatelja i primatelja u **red (*queue*)**
  - **Interni mehanizam**, može ih biti N u našem sustavu
  - Uklanja spregu (*coupling*) između pošiljatelja i primatelja
    - Dodatno, može: pohranjivati poruke, transformirati, LB, …
    - Tipično omogućuju pub/sub
  - Atributi: +FT, +HA, +S, -RT
  - Npr. Apache Kafka, RabbitMQ

# API Gateway (1)

- Kod rasta sustava često dolazi do razdvajanja API-ja u N tematski/logički grupiranih API-ja (servisa)
- ***API Gateway*** je alat za upravljanje API-jem koji se nalazi između klijenta i skupa pozadinskih usluga. Radi kao *reverse proxy* za prihvaćanje svih API poziva, agregiranje različitih usluga potrebnih za njihovo obavljanje i vraćanje odgovarajućeg rezultata.

https://www.redhat.com/en/topics/api/what-does-an-api-gateway-do

# API Gateway (2)

- Prednosti*:
  - Lakše interne izmjene
  - Sigurnost (autentifikacija, autorizacija) na jednom mjestu
  - Keširanje statičkog sadržaja
  - Nadziranje i upozoravanje (*monitoring and alerting*)
  - Protocol translation: npr. JSON – gRPC, HTTP-HTTPS, JSON-legacy XML, itd.
- Dobre prakse:
  - Ne smije sadržavati poslovnu logiku!
  - Može postati SPOF, ali možemo ih replicirati + LB
  - Ne zaobilaziti Gateway!
- Dodaje lagani RT overhead

* Software Architecture & Design of Modern Large Scale Systems, Michael Pogrebinsky

# Arhitekture web-aplikacija

… i još neki obrasci…

# *Multi-tier* (višeslojna) arhitektura

- Neprecizna terminologija na hr i eng:
  - ***Tier*** (fizička separacija) <> ***Layer*** (logička separacija softvera)
  - Nije nužno 1:1
- **Troslojna arhitektura** – najpopularnija!

# Troslojna arhitektura

- ✓ Dobra praksa - poslovna logika u aplikacijskom sloju
- ✓ Razvojni timovi razdvojeno raditi na slojevima (*tiers*)
- ✓ Možemo elegantno **skalirati** aplikacijski sloj (i ostale)
  - x Ali - skaliramo "sve ili ništa" ☹
- ✓ Dobra početna općenamjenska arhitektura

- x Temeljni nedostatak je **monolitna** struktura aplikacija u **aplikacijskom sloju**
- x **Prevelike cjeline**, „previše kôda”:
  - Teža podjela poslova među razvojnim timovima - organizacijska skalabilnost
  - Teže održavanje
  - Teže/sporije publiciranje novih verzija
    - „*overkill*” – sitna izmjena znači publiciranje kompletne aplikacije
    - što ako uvedemo pogrešku?

# Varijante

- 2T, 4T, …
- Izdvajanje servisa, API, API Gateway
- -> modularizacija

# Mikroservisna arhitektura

- Mikroservisna arhitektura strukturira aplikaciju kao skup usluga/servisa koje su:
  - Samostalno rasporedive (*indenedently deploy*)
  - Labavo spregnute (*loosely coupled*)
  - Organizirane oko poslovnih mogućnosti
  - U vlasništvu malog tima
  - Vrlo lako se održavaju i testiraju
  - Arhitektura mikroservisa omogućuje brzu, čestu i pouzdanu isporuku velikih, složenih aplikacija. Također omogućuje organizaciji da evoluira tehnologiju

https://microservices.io/

# Mikroservisna arhitektura

*A monolithic application puts all its functionality into a single process...*

*A microservices architecture puts each element of functionality into a separate service...*

*... and scales by replicating the monolith on multiple servers*

*... and scales by distributing these services across servers, replicating as needed.*

https://martinfowler.com/articles/microservices.html

# Mikroservisna arhitektura (3)

- Niz prednosti:
  - ✓ Skalabilnost
  - ✓ Organizacijska skalabilnost
  - ✓ Manji kod
  - ✓ FT - lakša izolacija pogrešaka
- Ali i mnogobrojni izazovi:
  - Nije lako ustrojiti i odrediti granice servisa
    - Single Responsibility Principle– there should not be multiple responsibilities in a microservice.
  - Database per service
  - *Big ball of mud*
  - ...

Sam Newman, author of *Building Microservices* and *Monolith to Microservices*, told attendees at the QCon developer conference in London that "microservices should not be the default choice.... **but a last resort**"

https://www.theregister.com/2020/03/04/microservices_last_resort/

# *Event driven* arhitektura (temeljena na događajima)

- Događaj (*event*) - maleni, samostalni, nepromjenjivi zapis nečega što se dogodilo
  - Sadrži opis događaja i trenutak događaja (event time)
  - Npr. dodavanje proizvoda u košaricu
- Omogućuje nam da smanjimo međusobnu ovisnost (mikro)servisa
- Naslanja se na *Message Brokere*
- Alterativno: Sage

# *Event sourcing*

- Alternativni način pohrane podataka – svaka **promjena** se pohranjuje kao **nepromjenjivi zapis – *event***.
  - *Model information about activity in the domain as a series of discrete events. Represent each event as domain object.*

    *Eric Evans, Domain-Driven Design Reference*
- Događaji se pohranjuju kao *append-only* niz, izmjena ili brisanje je zabranjeno ili se u najmanju ruku obeshrabruje
- Točnije: klijenti izdaju naredbe (***commands***) koje, ako uspješno prođu validaciju, postaju nepromjenjivi događaji – ***events***
  - Npr. kupnja karte, ako ima slobodnog mjesta, postaje event

- Prednosti: fleksibilno, omogućuje temporalne upite, objašnjava što se dogodilo (klasični pristup samo ažurira), reproducibilno, itd.
- Nedostatci: složenije, sporije, tipično u paru s CQRS obrascem i materijaliziranom read-only bazom, zahtjeva pouzdanu mrežnu infrastrukturu radi osjetljivosti na vrijeme, heterogeni eventi

# CQRS - *Command Query Responsibility Segregation*

- Razdvaja se čitanje i pisanje - dvije različite baze podataka:
  - Baza u koju pišemo propagira događaje pisanja
  - Koji se konsolidiraju u read-only bazu za čitanje

✓ Poboljšava skaliranje kad imamo **istovremeno visoke zahtjeve na čitanje i pisanje**

- Može se (i bolje) primijeniti samo na dio sustava

x Donosi značajnu dodatnu složenost

> *Despite these benefits, **you should be very cautious about using CQRS**. Many information systems fit well with the notion of an information base that is updated in the same way that it's read, adding CQRS to such a system can add significant complexity.*
>
> Martin Fowler

# *Serverless architecture*

- model u kojem **treći pružatelj (*3rd party*)** usluga (tipično u oblaku) sam brine o poslužitelju.
- Programeri moraju brinuti o kodu, a sve ostalo radi pružatelj poslužitelja:
  - ✓ sigurnosne zakrpe,
  - ✓ balansiranje opterećenja,
  - ✓ upravljanje kapacitetom,
  - ✓ skaliranje,
  - ✓ bilježenje i praćenje.
- Cijela ili dio aplikaciju na modelu bez poslužitelja
- "Prijatelji":
  - *Microservices* - **logička** organizacija/separacija koda
  - *Serverless* - **fizička**, način deploymenta, obavljanja

# Podatkovni sloj

# Podatkovni sloj

- Strategija skalabilnosti usmjerena na podatkovni sloj, tj. bazu podataka web aplikacije
- Dva glavna razloga za skaliranje baze podataka:
  1. prevelik broj pristupa bazi
     → replikacija
  2. prevelika količina podataka koje treba pohraniti
     → fragmentacija (*sharding, partitioning*)

- **Replikacija:** radi se u slučaju da ima puno zahtjeva za čitanje, a malo za pisanje – baza podataka se malo mijenja, a velik broj upita.
- Najčešće **MS**

# MS replikacija

- Kod **replikacije** podatke iz glavne baze podataka (*master/primary*) repliciramo u pomoćne baze podataka (*slave/secondary*).
- Prilikom svakog izmjene podataka u master, ažuriraju se i slave baze podataka.
- Upiti na baze se sada mogu raspodijeliti na više baza, što smanjuje opterećenje glavne baze. To ne povećava brzinu pisanja, ali glavna baza mora podržati zahtijevanu količinu pisanja.
- Možemo replicirati i replike. Problem koji se ovdje pojavljuje je kašnjenje replikacije (***replication lag***) u slučaju da se čita iz replicirane baze prije nego je podatak proslijeđen iz master baze.

# Particioniranje, fragmentiranje – *sharding*

- **Postupak particioniranja (*sharding*):** podjela baze podataka u disjunktni skup fragmenata koji obuhvaćaju sve podatke u bazi podataka uz zadovoljenje pravila da se baza podataka može rekonstruirati iz tih fragmenata bez gubitka informacije
- relacije mogu biti razdijeljene u fragmente horizontalno ili vertikalno (ili horizontalno i vertikalno)
- Više master računala:
  - U svaki *master* sprema se po dio podataka, raspodijeljenih po nekom algoritmu (hash, range, …)
  - Dodatna kompleksnost, usmjeravanje upita
  - Slabije podržano u relacijskim sustavima, nije lako zadržati ACID svojstva

# Fragmentacija & Replikacija su ortogonalni

# NoSQL sustavi

- NoSQL je široki pojam koji se koristi za označavanje raznih tipova pohrane podataka koji se razlikuju od tradicionalnog modela relacijskih baza
- NoSQL baze su više specijalizirane i pogodnije za skaliranje od tradicionalnih SQL baza
- Modeli podataka: *key-value, document, graph, column-family*
- Tipično podržavaju i replikaciju i fragmentaciju (izuzev graph)

# NoSQL vs RBP

RDB su najbolje ***one-size-fits-all*** rješenje

NoSQL su **specijalizirana** rješenja za određene (grupe) problema

# Polyglot persistance (1)

- Različite baze podataka su namjenjene za rješavanje različitih problema
- Hibridni pristup perzistenciji podataka: kombiniranje tehnologija kako bi se najbolje zadovoljile različite klase potreba za perzistencijom podataka
- Npr.

*NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence, Martin Fowler*

# *The Scale Cube* - tri dimenzije skaliranja

# Literatura

- Software Architecture & Design of Modern Large Scale Systems, Michael Pogrebinsky
- The Art Of Scalability, by Martin Abbott (Author), Michael Fisher (Author)

- Software Architecture in Practice (SEI Series in Software Engineering) by Len Bass , Paul Clements, et al
- Web Scalability for Startup Engineers, Tips & Techniques for Scaling Your Web Application, Artur Ejsmont

- Scalability Rules, Principles for Scaling Web Sites, Martin L. Abbott, Michael T. Fisher
- + reference u dokumentu

# Za one koji žele znati više…

… i dobiti par bodova više…

# Apache JMeter

# JMeter

- Apache JMeter testni alat omogućuje snimanje HTTP zahtjeva prema poslužiteljima i testiranje njihovih performansi
  - Desktop aplikacija otvorenog koda, besplatna, napisana u programskom jeziku Java
  - Provodi testove opterećenja (*load test, stress test*) koji su dio testa prihvaćanja (*acceptance test*) i potrebni prije puštanja aplikacije u produkciju
  - Performanse statističkih resursa (JavaScript i HTML) i dinamičkih (JSP, servleti, AJAX komponente)
  - Pomaže odrediti maksimalni broj istodobnih korisnika web aplikacije
  - Podržava izvještavanje, grafovi i izvješća o učinku (*performance reports*)
  - Preuzimanje: https://jmeter.apache.org/download_jmeter.cgi

# Izrada *Performance Test Plan* (1)

- Postupak izrade *Performance Test Plan*
- Pokrenuti aplikaciju:
  - jmeter.bat
  - ApacheJMeter.jar

# Izrada *Performance Test Plan* (2)

- Odabrati:
  - Add Thread Group
    - Number of Threads: 100
      - Broj simuliranih korisnika koji će povezati s web stranicama
    - Ramp-Up Period: 100
      - Koliko dugo će aplikacija čekati u sekundama prije pokretanja novog simuliranog korisnika. Primjerice, ako imamo 100 korisnika i Ramp-Up period je 100 s, onda će kašnjenje između pokretanja dva uzastopna korisnika biti 1 s (100 s /100 korisnika)
    - Loop Count: 10
      - Koliko puta će se test pokrenuti

# Izrada *Performance Test Plan* (3)

- Dodati JMeter elemente:
  - HTTP Request
    - U polje „Server Name or IP” unesite URL ili IP adresu poslužitelja koji testirate, npr. www.google.com
    - Unesite port na koji se šalju zahtjevi (80)
    - U polje „Path” unesite dodatne elemente zahtjeva koji šaljete na poslužitelja, npr. ako ovdje upišete „calendar” JMeter će izraditi zahtjev „http://www.google.com/calendar” i poslati ga na Google poslužitelj

# Izrada *Performance Test Plan* (4)

- Dodati izvještaje - vizualizacija rezultata testiranja
  - Odabrati „Graph Results” u meniju „Listener”
  - Podržano je više vrsta izvještavanja
  - Pokrenuti test („Run Test”) (ili CTRL+R)
  - Rezultati se prikazuju u stvarnom vremenu

Graf prikazuje rezultat plana testiranja u kojemu je simulirano 100 korisnika koji pristupaju web stranicama www.google.com

# Izrada *Performance Test Plan* (5)

- Analiza rezultata testa
  - Rezultati su predstavljeni u sljedećim bojama:
    - Crna: Ukupan broj poslanih zahtjeva
    - Plava: Trenutni prosjek svih poslanih zahtjeva
    - Crvena: Trenutna standardna devijacija
    - Zelena: Stopa propusnosti (*throughput rate*) koja predstavlja broj svih zahtjeva u minuti koje je poslužitelj obradio
  - Za analizu test opterećenja osobito su važne dvije vrijednosti:
    - Propusnost (*throughput*)
    - Devijacija (*deviation*)
  - Propusnost predstavlja sposobnost poslužitelja da podnosi visoku razinu opterećenja zahtjevima. Viša vrijednost je bolja.
  - U ovom testu propusnost Google poslužitelja bio je 582.247/minute, tj. toliko zahtjeva je poslužitelj mogao obraditi u 1 minuti.
  - Devijacija je u crvenoj boji (35). Manja vrijednost je bolja.

- Isti test može se pokrenuti više puta, rezultati će se agregirati

# View Results Tree i Summary Report

- Mogućnost View Results Tree
  - Vidljivi su detalji pojedinih HTTP zahtjeva
- Summary Report
  - Završni tablični izvještaj

# Ostale aplikacije

- Selenium WebDriver – aplikacija za automatizaciju web preglednika (automatsko upravljanje), W3C preporuka, https://www.selenium.dev/documentation/webdriver/
- Postman – izrada i ispitivanje performansi aplikacijskih programskih sučelja, https://www.postman.com/
- Pact – ispitivanje integriteta HTTP poruka (*contract testing*), npr. za mikroservisne arhitekture, https://docs.pact.io/

# API

API are a *contract that we define for other applications* so they car use our system without having to know anything about its *interna design and implementation*

- ~~Ugovor~~
- ~~Tri grupe~~
  - ~~Public, dobra praksa da se registriraju~~
  - ~~Partner~~
  - ~~Private, interno, unutar kompanije~~
  - ~~...~~
  - ~~Good patterns:~~
    - ~~1 complete encapsulation~~
    - ~~2 easy to use, understant~~
    - ~~3. keeping the operations idempotent~~
    - ~~4 api pagination, primjer email~~
    - ~~5 async operations, eg big data report~~
      - ~~Async api - odgovor odmah~~
    - ~~6. versioning api~~
    - ~~...~~

# RPC

- ~~Local transparency~~
- ~~From interface: Stubs: server and client~~
- ~~Around for decades…~~
- ~~Pick the framework…~~
- ~~Benefits:~~
  - ~~Convinience, from any lang~~
- ~~Drawback:~~
  - ~~Slow -> introduce async ops for slow methods~~
  - ~~Unreliability Primjer kreditna kartica - idempotentnost~~
- ~~When to use: najčešće server-to-server~~

# REST API

- ~~Not a standard or a protocol but style~~
- ~~Vs RPC, REST is resource oriented~~

–

- ~~How to achieve reqs:~~
  - ~~Statelesness~~
  - ~~Cacheability~~
- ~~Named resources~~
  - ~~Simple, singular~~
  - ~~Collection, plural names~~
  - ~~Best practice: nouns only, meaningful names, ids should be uniique and url friendly~~